The GENIE Neutrino Monte Carlo Generator

PHYSICS & USER MANUAL

C.Andreopoulos[a], F.Cavanna[b], J.Dobson[c], S.Dytman[d], H.Gallagher[e], P.Guzowski[c], R.Hatcher[f], Y.Hayato[g], P.Kehayias[e], A.Meregaglia[h], D.Naples[d], G.Pearce[a], A.Rubbia[i] and M.Whalley[j]

[a]Rutherford Appleton Laboratory, STFC, Oxfordshire OX11 0QX, UK

[b] Physics Dept., L'Aquila University, 67100 L'Aquila, Italy

[c]Physics Dept., Imperial College London, London SW7 2BW, UK

[d]Physics Dept., Pittsburgh University, Pittsburgh PA 15260, USA

[e]Physics Dept., Tufts University, Medford MA, 02155, USA

[f]Fermi National Accelerator Laboratory, Batavia, Illinois 60510, USA

[g]Kamioka Observatory, ICRR, Tokyo University, Kamioka, Gifu 506-1205, Japan

[h]IPHC - Strasbourg, F-67037 Strasbourg Cedex 2, France

[i]Physics Dept., ETH Zurich, CH-8093 Zurich, Switzerland

[j]Physics Dept., Durham University, Durham DH1 3LE, UK

(the GENIE collaboration)

October 12, 2009

Contents

Preface

GENIE [1] is a new neutrino event generator for the experimental neutrino physics community. The goal of the project is to develop a ‘canonical’ neutrino interaction physics Monte Carlo whose validity extends to all nuclear targets and neutrino flavors from MeV to PeV energy scales.

Chapter 1

Introduction

1.1 GENIE project overview

Over the last few years, throughout the field of high energy physics (HEP), we have witnessed an enormous effort committed to migrating many popular procedural Monte Carlo Generators into their C++ equivalents designed using object-oriented methodologies. Well-known examples are the GEANT [2], HERWIG [3] and PYTHIA [4] Monte Carlo Generators. This reflects a radical change in our approach to scientific computing. Along with the eternal requirement that the modeled physics be correct and extensively validated with external data, the evolving nature of computing in HEP has introduced new requirements. These requirements relate to the way large HEP software systems are developed and maintained, by wide geographically-spread collaborations over a typical time-span of $\sim$25 years during which they will undergo many (initially unforeseen) extensions and modifications to accommodate new needs. This puts a stress on software qualities such re-usability, maintainability, robustness, modularity and extensibility. Software engineering provides many well proven techniques to address these requirements and thereby improve the quality and lifetime of HEP software. In neutrino physics, the requirements for a new physics generator are more challenging for three reasons: the lack of a 'canonical' procedural generator, theoretical and phenomenological challenges in modeling few-GeV neutrino interactions, and the rapidly evolving experimental and theoretical landscape.

The long-term goal of this project is for GENIE to become a 'canonical' neutrino event generator whose validity will extend to all nuclear targets and neutrino flavors over a wide spectrum of energies ranging from $\sim$1 MeV to $\sim$1 PeV. Currently, emphasis is given to the few-GeV energy range, the challenging boundary between the non-perturbative and perturbative regimes which is relevant for the current and near future long-baseline precision neutrino experiments using accelerator-made beams. The present version provides comprehensive neutrino interaction modelling in the energy from, approximately, $\sim$100 MeV to a few hundred GeV.

GENIE[1] is a ROOT-based [5] Neutrino MC Generator. It was designed using

[1]**GENIE** stands for **G**enerates **E**vents for **N**eutrino **I**nteraction **E**xperiments

object-oriented methodologies and developed entirely in C++ over a period of more than three years, from 2004 to 2007. Its first official physics release (v2.0.0) was made available on August 2007. GENIE has already being adopted by the majority of neutrino experiments, including those as the JPARC and NuMI neutrino beamlines, and will be an important physics tool for the exploitation of the world accelerator neutrino program.

The project is supported by a group of physicists from all major neutrino experiments operating in this energy range, establishing GENIE as a major HEP event generator collaboration. Many members of the GENIE collaboration have extensive experience in developing and maintaining the legacy Monte Carlo Generators that GENIE strives to replace, which guarantees knowledge exchange and continuation. The default set of physics models in GENIE have adiabatically evolved from those in the NEUGEN [6] package, which was used as the event generator by numerous experiments over the past decade.

1.2 Neutrino Interaction Simulation: Challenges and Significance

Neutrinos have played an important role in particle physics since their discovery half a century ago. They have been used to elucidate the structure of the electroweak symmetry groups, to illuminate the quark nature of hadrons, and to confirm our models of astrophysical phenomena. With the discovery of neutrino oscillations using atmospheric, solar, accelerator, and reactor neutrinos, these elusive particles take center stage as the objects of study in and of themselves. Precision measurements of the lepton mixing matrix, the search for lepton CP violation, and the determination of the neutrino masses and hierarchy will be major efforts in HEP for several decades. The cost of this next generation of experiments will be significant, typically tens to hundreds of millions of dollars. A comprehensive, thoroughly tested neutrino event generator package plays an important role in the design and execution of these experiments, since this tool is used to evaluate the feasibility of proposed projects and estimate their physics impact, make decisions about detector design and optimization, analyze the collected data samples, and evaluate systematic errors. With the advent of high-intensity neutrino beams from proton colliders, we have entered a new era of high-statistics, precision neutrino experiments which will require a new level of accuracy in our knowledge, and simulation, of neutrino interaction physics [7].

While object-oriented physics generators in other fields of high energy physics were evolved from well established legacy systems, in neutrino physics no such 'canonical' MC exists. Until quite recently, most neutrino experiments developed their own neutrino event generators. This was due partly to the wide variety of energies, nuclear targets, detectors, and physics topics being simulated. Without doubt these generators, the most commonly used of which have been GENEVE [8], NEUT [9], NeuGEN [6], NUANCE [10] and NUX [11], played an important role in the design and exploration of the previous and current generation of accelerator neutrino experiments. Tuning on the basis of unpublished data from each group's own experiment has not been unusual making it virtually impossible to perform a global, independent evaluation for the state-of-the-art in neutrino interaction physics simulations. Moreover, limited manpower and the fragility of the overextended software architectures meant that many of these legacy physics generators were not keeping up with the latest theoretical ideas and experimental measurements.

Simulating neutrino interactions in the energy range of interest to current and near-future experiments poses significant challenges. This broad energy range bridges the perturbative and non-perturbative pictures of the nucleon and a variety of scattering mechanisms are important. In many areas, including elementary cross sections, hadronization models, and nuclear physics, one is required to piece together models with different ranges of validity in order to generate events over all of the available phase space. This inevitably introduces challenges in merging and tuning models, making sure that double counting and discontinuities are avoided. In addition there are kinematic regimes which are outside the stated range of validity of all available models, in which case we are left with the challenge of developing our own models or deciding which model best extrapolates into this region. An additional fundamental problem in this energy range is a lack of data. Most simulations have been tuned to bubble chamber data taken in the 70's and 80's. Because of the limited size of the data samples (important exclusive channels might only contain a handful of events), and the limited coverage in the space of ($\nu/\bar{\nu}$, E_ν, A), substantial uncertainties exist in numerous aspects of the simulations.

The use cases for GENIE are also informed by the experiences of the developers and users of the previous generation of procedural codes. Dealing with these substantial model uncertainties has been an important analysis challenge for many recent experiments. The impact of these uncertainties on physics analyses have been evaluated in detailed systematics studies and in some cases the models have been fit directly to experimental data to reduce systematics. These 'downstream' simulation-related studies can often be among the most challenging and time-consuming in an analysis.

To see the difficulties facing the current generation of neutrino experiments, one can look no further than the K2K and miniBoone experiments. Both of these experiments have measured a substantially different Q^2 distribution for their quasielastic-like events when compared with their simulations, which involve a standard Fermi Gas model nuclear model [12, 13]. The disagreement between nominal Monte Carlo and data is quite large - in the lowest Q^2 bin of miniBoone the deficit in the data is around 30\% [13]. It seems likely that the discrepancies seen by both experiments have a common origin. However the two experiments have been able to obtain internal consistency with very different model changes - the K2K experiment does this by eliminating the CC coherent contribution in the Monte Carlo [14] and the miniBoone experiment does this by modifying certain parameters in their Fermi Gas model [13]. Another example of the rapidly evolving nature of this field is the recently reported excess of low energy electron-like events by the miniBoone collaboration [15]. These discrepancies have generated significant new theoretical work on these topics over the past several years [16, 17, 18, 19, 20, 21, 22, 23]. The situation is bound to become even more interesting, and complicated, in the coming decade, as new high-statistics experiments begin taking data in this energy range. Designing a software system that can be responsive to this rapidly evolving experimental and theoretical landscape is a major challenge.

In this paper we will describe the ways in which the GENIE neutrino event generator addresses these challenges. These solutions rely heavily on the power of modern software engineering, particularly the extensibility, modularity, and flexibility of object oriented design, as well as the combined expertise and experience of the collaboration with previous procedural codes.

Chapter 2

Neutrino Interaction Physics

2.1 Medium Energy Range (100 MeV - 500 GeV)

2.1.1 Introduction

The set of physics models used in GENIE incorporates the dominant scattering mechanisms from several MeV to several hundred GeV and are appropriate for any neutrino flavor and target type. Over this energy range, many different physical processes are important. These physics models can be broadly categorized into nuclear physics models, cross section models, and hadronization models.

Substantial uncertainties exist in modeling neutrino-nucleus interactions, particularly in the few-GeV regime which bridges the transition region between perturbative and non-perturbative descriptions of the scattering process. For the purposes of developing an event generator this theoretical difficulty is compounded by the empirical limitation that previous experiments often did not publish results in these difficult kinematic regions since a theoretical interpretation was unavailable.

In physics model development for GENIE we have been forced to pay particular attention to this 'transition region', as for few-GeV experiments it dominates the event rate. In particular the boundaries between regions of validity of different models need to be treated with care in order to avoid theoretical inconsistencies, discontinuities in generated distributions, and double-counting. In this brief section we will describe the models available in GENIE and the ways in which we combine models to cover regions of phase space where clear theoretical or empirical guidance is lacking.

2.1.2 Cross section model

The cross section model provides the calculation of the differential and total cross sections. During event generation the total cross section is used together with the flux to determine the energies of interacting neutrinos. The cross sections for specific processes are then used to determine which interaction type occurs, and the differential distributions for that interaction model are used to determine the event kinematics. While the differential distributions must be calculated event-by-event, the total cross sections can be pre-calculated and stored for use by many jobs sharing the same physics models. Over this energy range neutrinos can scatter off a variety of different ‘targets’ including the nucleus (via coherent scattering), individual nucleons, quarks within the nucleons, and atomic electrons.

Quasi-Elastic Scattering: Quasi-elastic scattering (e.g. $\nu_\mu + n \to \mu^- + p$) is modeled using an implementation of the Llewellyn-Smith model [24]. In this model the hadronic weak current is expressed in terms of the most general Lorentz-invariant form factors. Two are set to zero as they violate G-parity. Two vector form factors can be related via CVC to electromagnetic form factors which are measured over a broad range of kinematics in electron elastic scattering experiments. Several different parametrizations of these electromagnetic form factors including Sachs [25], BBA2003 [26] and BBBA2005 [27] models are available with BBBA2005 being the default. Two form factors - the pseudo-scalar and axial vector, remain. The pseudo-scalar form factor is assumed to have the form suggested by PCAC, which leaves the axial form factor $\mathrm{F}_A(\mathrm{Q}^2)$ as the sole remaining unknown quantity. $\mathrm{F}_A(0)$ is well known from measurements of neutron beta decay and the Q^2 dependence of this form factor can only be determined in neutrino experiments and has been the focus of a large amount of experimental work over several decades. In GENIE a dipole form is assumed, with the axial vector mass m_A remaining as the sole free parameter with a default value of 0.99 GeV/c^2. For nuclear targets a suppression factor is being included from an analytic calculation of the rejection factor in the Fermi Gas model, based on the simple requirement that the momentum of the outgoing nucleon exceed the fermi momentum k_F for the nucleus in question. Typical values of k_F are 0.221 GeV/c for nucleons in ^{12}C, 0.251 GeV/c for protons in ^{56}Fe, and 0.256 GeV/c for neutrons in ^{56}Fe.

Elastic Neutral Current Scattering: Elastic neutral current processes are computed according to the model described by Ahrens et al. [28], including the strange quark contribution to the axial form factor. For nuclear targets the same reduction factor described above is used.

Baryon Resonance Production: The production of baryon resonances in neutral and charged current channels is included with Rein-Sehgal model [29]. This model employs the Feynmann-Kislinger-Ravndal [30] model of baryon resonances, which give wavefunctions for the resonances as excited states of a 3-quark system in a relativistic harmonic oscillator potential with spin-flavor symmetry. In the Rein-Sehgal paper the helicity amplitudes for the FKR model are computed and used to construct the cross sections for neutrino-production of the baryon resonances. From the 18 resonances of the original paper we include the 16 that are listed as unambiguous at the latest PDG baryon tables and all resonance parameters have been updated. For tau neutrino charged current interactions an overall correction factor to the total cross section is applied to account for neglected form factors in the original model. In our implementation of the Rein-Sehgal model interference between neighboring resonances has been ignored.

Coherent Neutrino-Nucleus Scattering: Coherent scattering results in the production of forward going pions in both charged current ($\nu_\mu + A \to \mu^- + \pi^+ + A$) and neutral current ($\nu_\mu + A \to \nu_\mu + \pi^0 + A$) channels. Coherent neutrino-nucleus interactions are modeled according to the Rein-Sehgal model [31]. Since the coherence condition requires a small momentum transfer to the target nucleus, it is a low-Q^2 process which is related via PCAC to the pion field.

The Rein-Sehgal model begins from the PCAC form at $Q^2=0$, assumes a dipole dependence for non-zero Q^2 and then calculates the pion-nucleus scattering cross section using various nuclear physics assumptions. The GENIE implementation is using the modified PCAC formula described in a recent revision of the Rein-Sehgal model [32] that includes lepton mass terms.

Non-Resonance Inelastic Scattering: Deep (and not-so-deep) inelastic scattering (DIS) is calculated in an effective leading order model using the modifications suggested by Bodek and Yang [33] to describe scattering at low Q^2. In this model higher twist and target mass corrections are accounted for through the use of a new scaling variable and modifications to the low Q^2 parton distributions. The cross sections are computed at a fully partonic level (the $\nu q \rightarrow l q'$ cross sections are computed for all relevant sea and valence quarks). The longitudinal structure function is taken into account using the Whitlow R ($R = F_L/2xF_1$) parameterization [34]. An overall scale factor of 1.032 is applied to the predictions of the Bodek-Yang model to achieve agreement with the measured value of the cross section at high energy.

Quasi-Elastic Charm Production: QEL charm production is modeled according to the Kovaleno local duality inspired mode [35] tuned by the GENIE authors to recent NOMAD data [36].

Deep-Inelastic Charm Production: DIS charm production is modeled according to the Aivazis, Olness and Tung model [37].

Inclusive Inverse Muon Decay: Inclusive Inverse Muon Decay cross section is computed using an implementation of the Bardin and Dokuchaeva model [38] taking into account all 1-loop radiative corrections.

Neutrino-Electron Elastic Scattering: The cross sections for all $\nu e-$ scattering channels other than Inverse Muon Decay is computed according to [39]. Inverse Tau decay is neglected.

2.1.3 Modeling the transition region

As discussed, for example, by Kuzmin, Lyubushkin and Naumov [40] one typically considers the total νN CC scattering cross section as

$$\sigma^{tot} = \sigma^{QEL} \oplus \sigma^{1\pi} \oplus \sigma^{2\pi} \oplus ... \oplus \sigma^{1K} \oplus ... \oplus \sigma^{DIS}$$

In the absence of a model for exclusive inelastic multi-particle neutrinoproduction, the above is usually being approximated as

$$\sigma^{tot} = \sigma^{QEL} \oplus \sigma^{RES} \oplus \sigma^{DIS}$$ [1]

assuming that all exclusive low multiplicity inelastic reactions proceed primarily through resonance neutrinoproduction. In this picture, one should be careful in avoiding double counting the low multiplicity inelastic reaction cross sections.

In GENIE release the total cross sections is constructed along the same lines, adopting the procedure developed in NeuGEN [6] to avoid double counting. The total inelastic differential cross section is computed as

$$\frac{d^2\sigma^{inel}}{dQ^2dW} = \frac{d^2\sigma^{RES}}{dQ^2dW} + \frac{d^2\sigma^{DIS}}{dQ^2dW}$$

[1] For the sake of formula compactness we omit the small contributions to the total cross section, in the few GeV energy range, from coherent scattering, elastic νe^- scattering etc.

The term $d^2\sigma^{RES}/dQ^2dW$ represents the contribution from all low multiplicity inelastic channels proceeding via resonance production. This term is computed as

$$\frac{d^2\sigma^{RES}}{dQ^2dW} = \sum_k \left(\frac{d^2\sigma^{R/S}}{dQ^2dW}\right)_k \cdot \Theta(Wcut - W)$$

where the index k runs over all baryon resonances taken into account, W_{cut} a configurable parameter and $(d^2\sigma^{RS}_{\nu N}/dQ^2dW)_k$ is the Rein-Seghal model prediction for the k^{th} resonance cross section.

The DIS term of the inelastic differential cross section is expressed in terms of the differential cross section predicted by the Bodek-Yang model appropriately modulated in the "resonance-dominance" region $W < W_{cut}$ so that the RES/DIS mixture in this region agrees with the cross section data for low multiplicity exclusive inelastic reactions:

$$\begin{aligned} \frac{d^2\sigma^{DIS}}{dQ^2dW} &= \frac{d^2\sigma^{DIS,BY}}{dQ^2dW} \cdot \Theta(W - Wcut) + \\ &+ \frac{d^2\sigma^{DIS,BY}}{dQ^2dW} \cdot \Theta(Wcut - W) \cdot \sum_m f_m \end{aligned}$$

In the above expression, m refers to the multiplicity of the hadronic system and, therefore, the factor f_m relates the total calculated DIS cross section to the DIS contribution to this particular multiplicity channel. These factors are computed as $f_m = R_m \cdot P_m^{had}$ where R_m is a tunable parameter and P_m^{had} is the probability, taken from the hadronization model, that the DIS final state hadronic system multiplicity would be equal to m. The approach described above couples the GENIE cross section and hadronic multiparticle production model [41]. The current default values for transition region parameters are W_{cut}=1.7 GeV/c^2, $R_2(\nu p) = R_2(\overline{\nu} n)$=0.1, $R_2(\nu n) = R_2(\overline{\nu} p)$=0.3, and R_m=1.0 for all $m > 2$ reactions.

The GENIE default charged current cross sections for quasi-elastic, single pion, and inclusive scattering from an isoscalar target, together with the estimated uncertainty on the total cross section, are shown in Fig. 2.1.

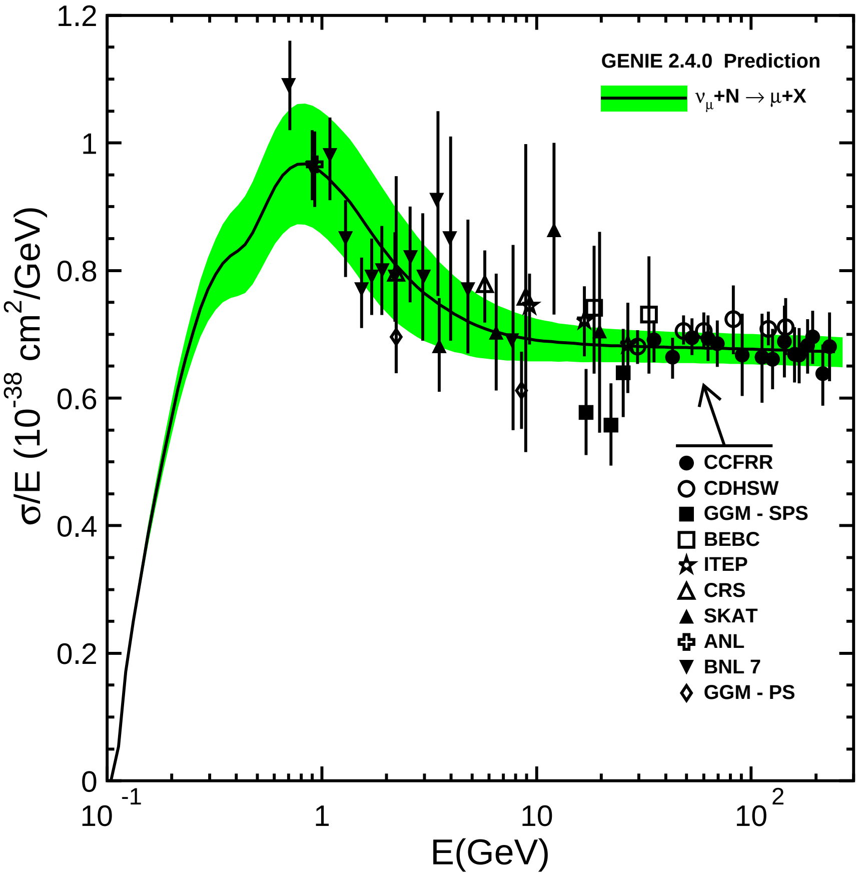

Figure 2.1: GENIE default cross sections for charged current scattering from an isoscalar target. The shaded band indicates the estimated uncertainty on the free nucleon cross section.

2.1.4 Neutrino-Induced Hadronic Multiparticle Production

Neutrino-induced hadronic shower modeling is an important aspect of the intermediate energy neutrino experiment simulations, as non-resonant inelastic scattering becomes the dominant interaction channel for neutrino energies as low as 1.5 GeV.

Experiments are sensitive to the details of hadronic system modeling in many different ways. Experiments making calorimetric measurements of neutrino energy in charged current reactions are typically calibrated using single particle test beams, which introduces a model dependence in determining the conversion between detector activity and the energy of neutrino-produced hadronic systems. Physics analysis can also depend on the prediction of the hadron shower characteristics, such as shower shapes, energy profile and particle content, primarily for event identification. A characteristic example is a $\nu_\mu \to \nu_e$ appearance analysis, where the evaluation of backgrounds coming from NC events, would be quite sensitive on the details of the NC shower simulation and specifically the π^0 shower content. It is therefore imperative that the state-of-the-art in shower modeling is included in our neutrino interaction simulations.

GENIE uses the AGKY hadronization model which was developed for the MINOS experiment [41]. This model integrates an empirical low-invariant mass model with PYTHIA/JETSET at higher invariant masses. The transition between these two models takes place over an adjustable window with the default values of 2.3 GeV/c^2 to 3.0 GeV/c^2, so as to ensure continuity of all simulated observables as a function of the invariant mass. For the hadronization of low-mass states the model proceeds in two phases, first determining the particle content of the hadronic shower, and secondly determining the 4-momenta of the produced particles in the hadronic center of mass.

The AGKY's low mass hadronization model generates hadronic systems that typically consist of exactly one baryon (p or n) and any number of π^+, π^-, π^0, K^+, K^-, K^0, $\bar{K^0}$ mesons kinematically possible and allowed by charge conservation.

For a fixed hadronic invariant mass and initial state (neutrino and hit nucleon), the algorithm for generating the hadron shower particles generally proceeds as follows:

- Compute the average charged hadron multiplicity using empirical expressions of the ($< n_{ch} >= a_{ch} + b_{ch} * lnW^2$) form. The coefficients a_{ch}, b_{ch}, which depend on the initial state, have been determined by bubble chamber experiments and are treated as tuning parameters.

- Compute the average hadron multiplicity as $< n_{tot} >= 1.5 < n_{ch} >$.

- Using the calculated average hadron multiplicity, generate the actual hadron multiplicity taking into account that the multiplicity dispersion is described by the KNO scaling law, ($< n > P(n) = f(n/ < n >)$) [42], where f is the scaling function. The KNO scaling is parametrized employing the Levy [2] function with an input parameter c_{ch} that depends on the initial state and is treated as a tuning parameter.

[2]The Levy function $Levy(z; c) = 2e^{-c}c^{cz+1}/\Gamma(cz + 1)$

- Generate hadrons up the generated hadron multiplicity taking into account the hadron shower charge conservation and the kinematical constraints. Protons and neutrons are produced in the ratio 2:1 for νp interactions, 1:1 for νn and $\bar{\nu}p$, and 1:2 for $\bar{\nu}n$ interactions. Charged mesons are then created in order to balance charge, and the remaining mesons are generated in neutral pairs. The probabilities for each are 31.33% (π^0, π^0), 62.66% (π^+, π^-), 1.5% (K^0, K^-), 1.5% (K^+, K^-), 1.5% $(\bar{K^0}, K^+)$ and 1.5% $(K^0, \bar{K^0})$. The probability of producing a strange baryon via associated production is determined from a fit to Λ production data:

$$P_{hyperon} = a_{hyperon} + b_{hyperon} \ln W^2 \tag{2.1}$$

Fig. 2.2 shows the data/model comparisons of the average charged hadron multiplicity $< n_{ch} >$ shown as a function of the squared hadronic invariant mass for
νp and νn interactions.

The main dynamical feature observed in the study of hadronic systems is that the baryon tends to go backwards in the hadronic center of mass and that the produced hadrons have small transverse momentum relative to the direction of momentum transfer. These features are naturally described in the quark model where the baryon is formed from the diquark remnant, which goes backwards in the center-of-mass, and transverse momentum is generated solely through intrinsic parton motion and gluon radiation. At low invariant masses energy-momentum constraints on the available phase space play a particularly important role. The most pronounced kinematical feature in this region is that one of the produced particles (proton or neutron) is much heavier that the rest (pion and kaons) and exhibits a strong directional anticorrelation with the momentum transfer.

Our strategy, therefore, is to correctly reproduce the final state nucleon momentum, and then perform a phase space decay on the remnant system employing, in addition, a p_T-based rejection scheme designed to reproduce the expected meson transverse momentum distribution. The nucleon momentum is generated using input p_T^2 and x_F PDFs which are parametrized, based on experimental data [43, 44]. Once the baryon momentum is selected the remaining particles undergo a phase space decay. The phase space decay employs a rejection method suggested in [45], with a rejection factor e^{-A*p_T} for each meson. This causes the transverse momentum distribution of the generated mesons to fall exponentially with increasing p_T^2. Here p_T is the momentum component perpendicular to the current direction.

Fig. 2.4 shows the data/model comparisons of the fragmentation function for positively and negatively charged hadrons. 2-body hadronic systems are a special case: The hadronic system 4-momenta are generated by a simple unweighted phase space.

2.1.5 Nuclear Physics

Nuclear physics plays a large role in every aspect of neutrino scattering simulations at few-GeV energies and introduces coupling between several aspects of the simulation. The basic nuclear model used in GENIE is the Fermi Gas

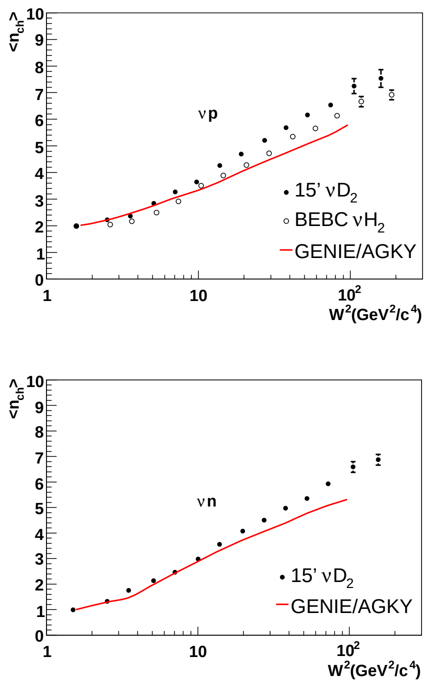

Figure 2.2: Data/model comparisons of the average charged hadron multiplicity $< n_{ch} >$ shown as a function of the squared hadronic invariant mass for νp and νn interactions.

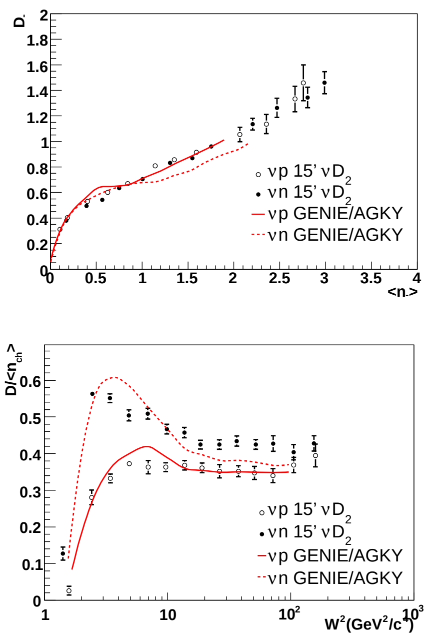

Figure 2.3: Data/model comparisons of the negatively charged hadron multiplicity dispersion D_- as a function of the average charged hadron multiplicity $< n_- >$ (up) and the reduced dispersion $D_-/ < n_- >$ as a function of the squared hadronic invariant mass (down).

Model, which has been modified by Bodek and Ritchie to incorporate nucleon-nucleon correlations [46]. This model is used to determine the Fermi motion of the initial target and to provide the prescription for treating the kinematics for scattering from an off-shell particle.

It is well known that scattering kinematics for nucleons in a nuclear environment are different from those obtained in scattering from free nucleons. For quasi-elastic and elastic scattering, Pauli blocking is applied as described in Sec. 2.1.2. For nuclear targets a nuclear modification factor is included to account for observed differences between nuclear and free nucleon structure functions which include shadowing, anti-shadowing, and the EMC effect [33].

Nuclear reinteractions of produced hadrons is simulated using a cascade Monte Carlo which will be described in more detail in the following section. The struck nucleus is undoubtedly left in a highly excited state and will typically de-excite by emitting nuclear fission fragments, nucleons, and photons. These photons are the particles most likely to be visible in a detector and are the only particles explicitly simulated.

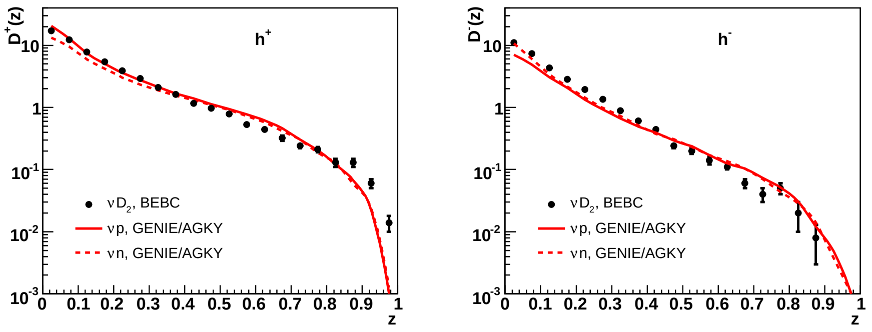

Figure 2.4: Data/model comparisons of the fragmentation function for positively and negatively charged hadrons.

2.1.6 Intranuclear Rescattering

The hadronization model describes particle production from free targets and is tuned primarily to bubble chamber data on hydrogen and deuterium targets. Hadrons produced in the nuclear environment may rescatter on their way out of the nucleus, and these reinteractions significantly modify the observable distributions in most detectors.

It is also well established that hadrons produced in the nuclear environment do not immediately reinteract with their full cross section. The basic picture is that during the time it takes for quarks to materialize as hadrons, they propagate through the nucleus with a dramatically reduced interaction probability. This was implemented in GENIE as a simple 'free step' at the start of the intranuclear cascade during which no interactions can occur. The formation time is the only free parameter and is 0.523 fm/c according to the SKAT model [47].

Intranuclear rescattering in GENIE is handled by a subpackage called INTRANUKE. INTRANUKE is an intranuclear cascade simulation and has gone through numerous revisions since the original version was developed for use by the Soudan 2 Collaboration [48]. The sensitivity of a particular experiment to intranuclear rescattering depends strongly on the detector technology, the energy range of the neutrinos, and the physics measurement being made. INTRANUKE simulates rescattering of pions and nucleons in the nucleus.

In principle one would like to have a fully realistic nuclear model which accurately describes the full range of nuclear physics down to 1 MeV to ensure that the simulations are suitable for any conceivable experiment. Nuclear simulations of this type are themselves highly complex packages and the integration of these packages with GENIE is an area of active work. An alternative approach is to develop a simpler nuclear model, in the context of a particular experiment, and ensure that the relevant physics for that experiment are correctly described. This approach has the advantage of yielding a far simpler code, which is understood by the experimenters. This has particular advantages for the study of systematic errors and the development of ancillary code like reweighting packages.

The current version was optimized for use by the MINOS experiment. For this experiment the task of developing an intranuclear rescattering model is simplified because the detector is composed largely of a single element, iron, and the detector is designed to make a calorimetric energy measurement rather than track individual particles. For the oscillation measurement of MINOS the primary goal is ensuring that the 'missing energy' lost in the nuclear environment is being reliably simulated.

The simulation tracks particles through the nucleus in steps of 0.1 fm. For each particle only one reinteraction is allowed, and the simulation consists of the following steps:

1. **Mean Free Path:** In order to determine if the particle interacts in a particular step the mean free path is calculated based on the local density. The mean free path is calculated from known hadron-nucleon cross sections [49] and known charge densities [50]. All nuclei heavier than oxygen are modeled with a Woods-Saxon density distribution and lighter nuclei are modeled with a modified Gaussian distribution:

   $$\rho(r) = \rho_0 \left[1 + \alpha \left(\frac{r}{a}\right)^2\right] e^{-r^2/a^2}. \tag{2.2}$$

   One difficulty in this approach is that our treatment is using a semiclassical model to describe a quantum mechanical process. This poses particular difficulty in describing elastic scattering which dominates the total cross section at low energy. This wave/particle distinction depends on energy with lower energy hadron-nucleus scattering being more wave-like. To account for this we increase the size of the nuclear density distribution through which the particle is tracked by an amount

   $$f\frac{hc}{p}, \tag{2.3}$$

   where f is an adjustable dimensionless parameter set to 0.5 in the current default. We use the isospin-averaged total cross sections for pions and nucleons and isospin relations for $\pi^0 - nucleon$ reactions.

2. **Determining the Interaction Type:** Hadron-nucleus interactions occur with different processes and each has an associated cross section - σ_{elas} for elastic scattering, σ_{inel} for inelastic scattering (which includes single nucleon emission), σ_{cex} for single charge exchange for all hadrons. For pions, emission of 2 or more nucleons with no pion in the final state is called absorption - σ_{abs}; for nucleons, a final state with 2 or more nucleons is called multi-nucleon knockout - σ_{ko}. The total cross section (σ_{tot}) is the sum of all component cross sections and the total reaction cross section (σ_{reac}) is the sum of all inelastic reactions,

   $$\sigma_{reac} = \sigma_{cex} + \sigma_{inel} + \sigma_{abs} = \sigma_{tot} - \sigma_{elas}. \tag{2.4}$$

   Once it has been determined that a hadron reinteracts in the nucleus, the type of the interaction is determined based on the measured cross sections for the above listed processes. In some cases, where data is sparse, cross section estimates are taken from estimates by the CEM03 group [51].

3. **Final State Products:** Once the interaction type has been determined, the four-vectors of final state particles need to be generated. Where possible these distributions are parametrized from data or from the output of

more sophisticated nuclear models [52]. For MINOS, the most important issue is missing energy generated by inelastic and absorption processes. Very low energy hadrons and nuclear recoils are not seen, so simplifications can be made. All states where more than 5 nucleons are emitted are treated as though 5 nucleons (3 protons and 2 neutrons) were emitted. The energy and momentum of the rescattered haadron are distributed among the final state nucleons according to phase space.

The intranuclear rescattering model has been tested and tuned based on comparisons to hadron-nucleus data. Hadron-nucleus cross sections are calculated by 'MC experiments' where a nucleus is being illuminated by a uniform hadron beam with transverse radius larger than the nucleus size. Figure 2.5 shows the comparison between INTRANUKE and data for π^+−Fe total and reaction cross sections. Validation of the model against neutrino data on nuclear targets have also been performed [48]. Although the model is tuned to hadron scattering on iron, the simplicity of the Fermi Gas model and the A$^{(2/3)}$ scaling of the cross sections allow the model to be applied to nearly all nuclei encountered in the simulation as well.

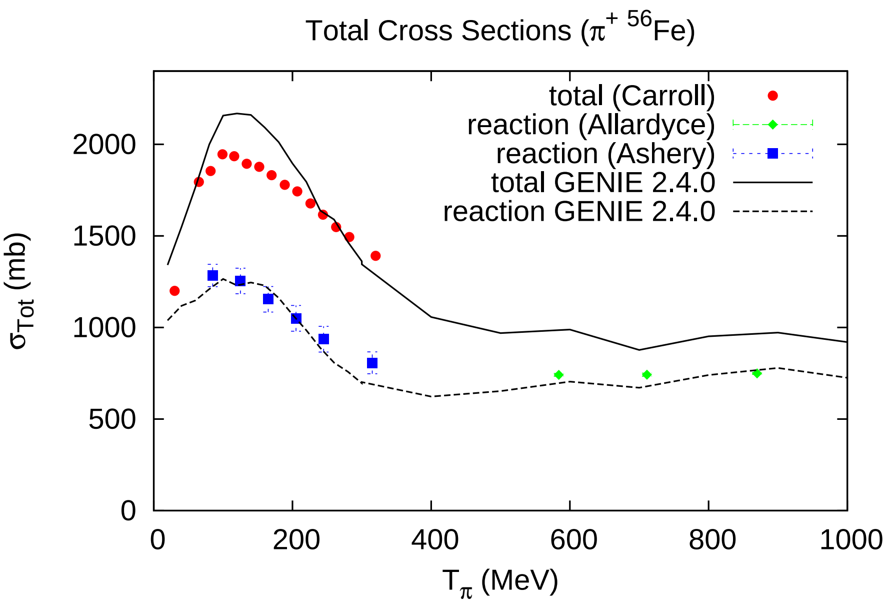

Figure 2.5: Pion scattering cross sections from iron.

2.2 The Low Energy Extension (10 MeV to 100 MeV)

[to be written]

2.3 The High Energy Extension ($>$ 500 GeV)

[to be written]

Chapter 3

Downloading & Installing GENIE

3.1 Understanding the versioning scheme

CVS Tags

In the GENIE version numbering scheme, releases are tagged in the CVS source code repository as ***R-major_minor_revision***[1]. When a number of significant functionality improvements or additions have been made, the major index is incremented. The minor index is incremented in case of significant fixes and/or minor feature additions. The revision number is incremented for minor bug fixes and updates.

Version number semantics

- Following the LINUX kernel versioning scheme, the versions with even minor number (eg 2.0.*, 2.4.*) correspond to stable, fully validated physics production releases[2].
- Versions with odd minor number (eg. 2.3.*, 2.5.*) correspond to the development version or 'candidate' releases tagged during the validation stage preceding a physics release.
- Tagged versions always have an even revision number (eg 2.2.2 is a revision of the 2.2.0 physics production release whereas 2.3.2, 2.3.4, 2.3.6,... are 'candidate' releases for the 2.4.0 physics production release).
- Odd revision numbers are used for the CVS head / development version.

Release codenames

The major production-quality releases are code-named after modern extinct or endangered species (series of production releases: ***Auk***, ***Blueback***, ***Cheetah***, ***Dodo***, ***Elk***, ***Fox***, ***Gazelle***, ***Hippo***, ***Ibex***,...).

[1]For example, tag R-1_99_1 corresponds to GENIE vrs 1.99.1, tag R-2_0_2 corresponds to GENIE vrs 2.0.2 etc.

[2]To the dismay of mathematicians, our versioning scheme uses 0 as am even number.

Release qualifiers

The GENIE releases are marked as:

- ***physics*** : Validated production-quality versions recommended for physics studies.
- ***depreciated***: Older 'physics' versions that have been greatly superseded by newer versions. Versions marked as 'depreciated' become unsupported. We appreciate that experiments get highly attached on specific versions due to the enormous amount of work invested in generating high statistics samples and calculating MC-dependent corrections and systematics. We strive to support as many physics versions as reasonably possible.
- ***pre-release***: Test releases you may not use for physics studies.
- ***special***: Releases prepared for a particular reason or event such as a) the evaluation of an experiment systematic with an appropriately modified version of GENIE, or b) a GENIE tutorial or a summer / winter school. You may not use these releases outside the intended context.

3.2 Obtaining the source code

The official GENIE source code is maintained at a CVS repository hosted at the Rutherford Appleton Laboratory[3].

Read-only access to GENIE CVS repository via AFS

To get the CVS head (development version), type:

```
shell$ cvs -d /afs/rl.ac.uk/user/c/candreop/cvsrep co -P GENIE
```

To get a specific tagged version you need to specify the CVS tag using the -r option. For example, in order to get version 2.4.2 (with a corresponding CVS tag: R-2_4_2), type:

```
shell$ cvs -d /afs/rl.ac.uk/user/c/candreop/cvsrep co -P -r R-2_4_2 GENIE
```

Read / write access to GENIE CVS repository via SSH

This is intended only for developers.

[3]A user heads-up is in order here: A Subversion (SVN) repository, hosted at the Rutherford Appleton Laboratory, is currently being tested. At some point, over the next few months, the GENIE source code will be migrated to the SVN repository and the CVS repository will be removed. The move will be announced at the GENIE user mailing list well in advance. SVN is a CVS successor. Please familiar yourself with SVN as the existing CVS repository may be discontinued at very short notice. More details on SVN can be found on: http://subversion.tigris.org.

3.3 Understanding prerequisites

Supported platforms

GENIE should build on all reasonably recent LINUX and MAC OS X distributions. A list of systems where GENIE has been successfully installed and used is maintained at the GENIE web site. As a matter of courtesy, please let us know whether you have GENIE successful installed at a new system.

External dependencies

A typical GENIE installation[4] requires the following external packages[5]:

- ***ROOT***
- ***LHAPDF***
  The Les Houches Accord PDF interface, a PDFLIB successor.
- ***PYTHIA6***
  The well known LUND Monte Carlo package used by GENIE for particle decays and string fragmentation (for neutrino interactions of high invariant mass).
- ***log4cpp***
  A C++ library for message logging.
- ***libxml2***
  The C XML library for the GNOME project.

More external packages are required to enable certain specialized features: You need MySQL to enable certain physics validation tools, and you need additional external MC generators to install alternative / non-default intranuclear cascade models. Doxygen and GraphViz is needed in order to generate your own copy of the doxygen documentation and google-perftools is needed for GENIE profiling. These are not typical user options and no further details are given in this manual.

Installing external dependencies

The installation of external packages is described in detail in their corresponding web pages. Additional detailed instructions can also be found at Appendix A ('Installation instructions for absolute beginners') of this manual. An abridged version of the installation instructions can be found below:

1. Install libxml2, log4cpp and LHAPDF first. Installation is completely straightforward and trivial. Pre-compiled binaries are also readily available. Note that libxml2 (almost certainly) and log4cpp (less likely) may already be installed at your system.

[4]A minimal installation that can be used for event generation / physics studies.

[5]The implicit assumption here is that you start with a 'working system' where some basic tools, such as the gcc compiler suite, make, autoconf, PERL, CVS and SVN clients etc, are already installed. Instructions are given assuming that you are using the bash shell but it is trivial to adapt these instructions for your own shell.

2. Install PYTHIA6. First you need to a) obtain the PYTHIA6 source code from [?], b) obtain the source code for a ROOT/PYTHIA thin wrapper library from the ROOT web site c) edit the PYTHIA6 source code and remove dummy parton density function calls and d) build the source code. The easiest way, by far, to do all the above is to use Robert Hatcher's *'build_pythia6.sh'* script. It automates getting all PYTHIA components from the web, removing dummy calls and building the code.

   A version of Robert's script is also kept at the GENIE web site at:
   http://hepunx.rl.ac.uk/∼candreop/generators/GENIE/etc/installation/

   You can run the script (please, also read its documentation) as:
   `shell$ source build_pythia6.sh [version]`

   For example, in order to download and install version 6.4.12, type:
   `shell$ source build_pythia6.sh 6412`

3. Install ROOT following the instructions at the ROOT web site. The only required non-default configuration options are the ones for enabling PYTHIA6 (Add: '`--enable-pythia6 --with-pythia6-libdir=/some/path`').

3.4 Preparing your environment

A number of environmental variables need to bee set or updated before using GENIE.

- Set the '`GENIE`' environmental variable to point at the top level GENIE directory
- Set the '`ROOTSYS`' environmental variable to point at the top level ROOT directory
- Set the '`LHAPATH`' environmental variable to point to LHAPDF's PDF data files
- Append '`$ROOTSYS`/*bin*' and '`$GENIE`/*bin*' to your '`PATH`'
- Append '`$ROOTSYS`/*lib*', '`$GENIE`/*lib*' and the paths to the log4cpp, libxml2, LHPADF and PYTHIA6 libraries to your '`LD_LIBRARY_PATH`' environmental variable (or to your '`DYLD_LIBRARY_PATH`' environmental variable if you are using GENIE on MAC OS X)

It is more convenient to create a GENIE setup script and execute it before using GENIE.
A setup script should look like the following:

```
#!/bin/bash

export GENIE=/path/to/genie/top/directory

export ROOTSYS=/path/to/root/top/directory
export LHAPATH=/path/to/lhapdf/PDFSets/
export PATH=$PATH:\
$ROOTSYS/bin:\
$GENIE/bin

export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:\
/path/to/log4cpp/library:\
/path/to/libxml2/library:\
/path/to/lhapdf/libraries:\
/path/to/pythia6/library:\
$ROOTSYS/lib:\
$GENIE/lib
```

Assuming that the above script is named '*genie_setup*', you can execute it by typing:

```
shell$ source genie_setup
```

3.5 Configuring GENIE

A configuration script is provided with the GENIE source code to help you configure your GENIE installation (enable / disable features and specify paths to external packages). To see what configuration options are available, type:

```
shell$ cd $GENIE
shell$ ./configure --help
```

This will generate a screen output that looks like the following:

```
FLAG              DESCRIPTION

--prefix          Install path (for make install)

enable/disable options
prefix with either --enable or --disable
(eg. --enable-lhapdf --disable-flux-drivers)

profiler           GENIE code profiling using Google perftools
doxygen-doc        Generate doxygen documentation at build time
dylibversion       Adds version number in dynamic library names
lowlevel-mesg      Disable (rather than filter out) some prolific
                   debug level messages known to slow GENIE down
debug              Adds -g compiler option to request debug info
neut-cascade       Alternative intranuclear cascade MC, requires NEUT
lhapdf             Use the LHAPDF library
cernlib            Use CERNLIB
flux-drivers       Built-in flux drivers
```

```
geom-drivers       Built-in detector geometry drivers
mueloss            Muon energy loss modeling
validation-tools   GENIE physics model validation tools
viewer             Event generation GUI
test               Test programs
t2k                T2K-specific ev/gen drivers and support softw.
minos              MINOS-specific ev/gen drivers and support softw.

with options for 3rd party software
prefix with --with (eg. --with-lhapdf-lib=/some/path)

optimiz-level      Compiler optimiz. level (O,O2,O3,OO,Os)
profiler-lib       Path to profiler library
doxygen-path       Path to doxygen binary
neut-lib           Path to NEUT libraries
pythia6-lib        Path to PYTHIA6 library
cern-lib           Path to CERN libraries
lhapdf-inc         Path to LHAPDF includes
lhapdf-lib         Path to LHAPDF libraries
libxml2-inc        Path to libxml2 includes
libxml2-lib        Path to libxlm2 library
log4cpp-inc        Path to log4cpp includes
log4cpp-lib        Path to log4cpp library
```

By default all options required for a minimal installation that can be used for physics event generation are enabled and non-essential features are disabled. Typically, the folowing should be sufficient for most users:

```
shell$ cd $GENIE
shell$ ./configure
```

Not specifying any configuration option (like above) is equivalent to specifying:

```
--disable-profiler
--disable-doxygen-doc
--enable-dylibversion
--disable-lowlevel-mesg
--disable-debug
--disable-neut-cascade
--disable-gibuu
--enable-lhapdf
--disable-cernlib
--enable-flux-drivers
--enable-geom-drivers
--enable-mueloss
--disable-validation-tools
--disable-viewer
--disable-t2k
--disable-minos
```

The default optimization level is set to O2 and `--prefix` is set to /usr/local.

The configuration script can, in principle, *auto-detect the paths* to required external packages installed at your system if no path is given explicitly. On some occasions, before scanning your system for external products, the configuration script will check whether some rather standard environmental variables have been set (from example, before searching for the PYTHIA6 / JETSET library, the configure script will check whether a '`PYTHIA6`' environmental variable has been set. See '`./configure --help`' for more information).

Obviously, if you want greater control over the configuration options (so that you do not depend on pre-set defaults that may one day change), if you want to modify some other default options or if the script fails to discover some external product path, then do set the configure script options explicitly.

3.6 Building GENIE

Once GENIE has been properly configured, you are ready to build it. Just type:

```
shell$ cd $GENIE
shell$ gmake
```

On successful completion you should be able to find many libraries located in *$GENIE/lib* and some applications and scripts in *$GENIE/bin.*

You may stop the building procedure here and start using GENIE now! However, some users may prefer to take their installation one step further and type:

```
shell$ gmake install
```

If */some/path* was the location specified via the `--prefix` configuration flag, then '`gmake install`' will:

- move all executables and scripts to */some/path/bin,*
- move all libraries to */some/path/lib,* and
- move all headers to */some/path/include/GENIE.*

If you do run '`gmake install`', before running GENIE you need to update your '`LD_LIBRARY_PATH`' (or '`DYLD_LIBRARY_PATH`' on MAC OS X) and '`PATH`' environmental variables accordingly.

Whether you stop the installation procedure after the '`gmake`' or '`gmake install`' step is probably more a matter of personal taste [6]. Whatever you choose should work given that your system's paths have been properly set.

Assuming now that the GENIE installation has been completed without apparent errors, we are going to provide instructions for a couple of simple post-installation tests to verify that GENIE has been properly built.

[6] I find it easier to manage multiple GENIE installations if I stop after the '`gmake`' step.

3.7 Performing simple post-installation tests

Here are few simple things you can do in order to try out your installation:

1. Compute the $\nu_\mu + C^{12}$ (ν_μ PDG code: 14, C^{12} PDG code: 1000060120) QEL cross section splines and then generate a 10,000 event sample of $\nu_\mu + C^{12}$ QEL interactions, between 0 and 15 GeV, using a simple histogram-based description of the T2K ν_μ flux (ROOT *TH1D* object 'h30000' stored in '*$GENIE/data/flux/t2kflux.root*').

   The commands used here will be explained in the next section:

   ```
   shell$ export GEVGL=QEL
   shell$ gmkspl -p 14 -t 1000060120 -n 300 -e 30 -o xnumuC12qel.xml
   shell$ export GSPLOAD=xnumuC12qel.xml
   shell$ gevgen -s -n 10000 -p 14 -t 1000060120 -e 0,15
   -f $GENIE/data/flux/t2kflux.root,h30000
   ```

   A '*genie-mcjob-0.status*' status file is created. It is updated periodicaly with job statistics and the most recent event dump. When the job is completed a '*gntp.0.ghep.root*' file, containing the generated event tree, is written-out. Print-out the first 100 eventsfrom this file:
   ```
   shell$ gevdump -f gntp.0.ghep.root -n 100
   ```

2. Generate a 10,000 event sample of $\pi^+ + O^{16}$interactions for π^+'s of 200 MeV kinetic energy.
   (π^+ PDG code: 211, O^{16} PDG code: 1000080160):

   ```
   shell$ ghAevgen -n 10000 -p 211 -t 1000080160 -k 0.2
   ```

If everything seems to work then the GENIE is really 'out of the bottle'. Continue reading the Physics and User Manual on how to run more involved MC jobs.

Chapter 4

Getting started with GENIE in neutrino mode

4.1 Introduction

[to be written]

4.2 Event generation inputs

4.2.1 Why using pre-calculated cross section splines

When generating neutrino interaction events, most CPU-cycles are spent on calculating neutrino interaction cross sections. In order to select an interaction channel for a neutrino scattered off a target at a particular energy, the differential cross section for each possible channel is integrated over the kinematic phase space available at this energy. With $\sim 10^2$ possible interaction modes per initial state and with $\sim 10^5$ differential cross section evaluations per cross section integration then $\sim 10^7$ differential cross section evaluations are required just in order to select an interaction channel for a given initial state. Had you been simulating events in a realistic detector geometry ($\sim 10^2$ different isotopes) then the number of differential cross section evaluations, before even starting simulating the event kinematics, would rise to $\sim 10^9$. It is therefore advantageous to pre-calculate the cross section data. The event generation drivers can be instructed to to load the pre-computed data and estimate the cross section by numerical interpolation, rather than by performing numerous CPU-intensive differential cross section integrations. The cross section data are written out in XML format and, when loaded into GENIE, they are used for instantiating *Spline* objects.

4.2.2 Generating your own splines

GENIE provides the *gmkspl* utility for generating the cross section splines for a specified set of modeled processes for a specified list of initial states. The cross section splines are written out in an XML file in the format expected by all other GENIE programs. The source code for this utility may be found in

'*$*GENIE*/src/stdapp/gMakeSplines.cxx*'.

In summary, the *gmkspl* syntax is:

```
shell$ gmkspl -p nu <-t tgt, -f geom> [-o out_file] [-n nknots] [-e Emax]
```

where:
[] marks optional arguments, and
<> marks a list of arguments out of which only one can be selected at any given time.

Details on the command-line arguments can be found below:

- **-p** Specifies the neutrino PDG codes.
  Multiple neutrino codes can be specified as a comma separated list.
- **-t** Specifies the target PDG codes.
  Multiple target PDG codes can be specified as a comma separated list.
- **-f** Specifies a ROOT file containing a ROOT/GEANT detector geometry description.
- **-o** Specifies the name of the output XML file
  By default GENIE writes-out the calculated cross section splines in an XML file named '*xsec_ splines.xml*' created at the current directory.
- **-n** Specifies the number of knots per spline
  By default GENIE is using 15 knots per decade of the spline energy range and at least 30 knots overall.
- **-e** Specifies the maximum neutrino energy in spline
  By default the maximum energy is set to be the declared upper end of the validity range of the event generation thread responsible for generating the cross section spline.

Examples:

1. To calculate the cross section splines for ν_μ (PDG code: 14) and $\bar{\nu_\mu}$ (PDG code: -14) scattered off Fe^{56} (PDG code: 1000260560), type:

   ```
   shell$ gmkspl -p 14,-14 -t 1000260560
   ```

   The cross section splines will be saved in an output XML file named *'xsec_ splines.xml* ' (default name).

2. To calculate the cross section splines for ν_μ (PDG code: 14) and $\bar{\nu_\mu}$ (PDG code: -14) scattered off all the targets in the input ROOT geometry file '*/data/mygeometry.root*' and write out the splines in a file named *'mysplines.xml*', type

   ```
   shell$ gmkspl -p 14,-14 -f /data/mygeometry.root -o mysplines.xml
   ```

Be warned that generating the cross section splines is very CPU-intensive because of the fine numerical integration stepping used in order to improve numerical accuracy and the large number of processes involved. See Fig. 4.1. Work is currently under way to improve the speed of the the numerical integration algorithms while retaining their numerical accuracy.

4.2.3 Downloading pre-computed splines

Cross section spline XML files are kept in:
http://hepunx.rl.ac.uk/∼candreop/generators/GENIE/data/cross-section-splines/

You need to select file corresponding to the version of GENIE you are using.

Typically I post cross section spline files for all modeled processes for ν_e, $\bar{\nu_e}$, ν_μ, $\bar{\nu_\mu}$, ν_τ, $\bar{\nu_\tau}$ scattered off free-nucleons (p, n) and off a large set of nuclear targets (the ~ 40 isotopes that can be found in the T2K detector geometries[1]). Using the posted free-nucleon cross section data is easy / fast to calculate cross section splines for any set of nuclear targets.

Any reasonable request for providing additional cross section splines will be satisfied.

4.2.4 Tricks for faster spline calculation

Cross section spline calculation is CPU-intensive. It is recommended that, for the default GENIE configuration, you use the spline XML files posted online.

If you decide to build your own splines for multiple neutrino species and nuclear targets, then it is impractical to generate all splines in a single job (although you can do so). It is much more practical to 'split' the cross section spline generation into many smaller jobs and run them on parallel at a batch farm. For faster results, organize your jobs as 'single neutrino + multiple nuclear targets' rather than 'multiple neutrinos + single nuclear target': In the former case intermediate, CPU-intensive free nucleon cross section calculations, for the given neutrino species, will be recycled in the nuclear target cross section calculations. The XML outputs of all the *gmkspl* jobs run in parallel can be merged into a single XML file using GENIE's *gspladd* utility. The *gspladd* utility syntax for is:

```
shell$ gspladd -i input1.xml,input2.xml,...  -o /path/output.xml
```

Using the free-nucleon cross section files posted at the GENIE web-site it is both easy and fast to calculate cross section splines for any set of nuclear targets. Because of the way nuclear effects are currently handled, the most CPU intensive cross section calculations can recycle free-nucleon calculations resulting in a dramatic speed improvement. The process of feeding-in free-nucleon cross section in a nuclear-cross section calculations is outlined below. Assuming that a complete set of free-nucleon cross section splines is stored at '*/path/freenuc.xml*'

[1]N^{14}, N^{15}, O^{16}, O^{17}, O^{18}, Al^{27}, C^{12}, C^{13}, H^2, Cl^{35}, Cl^{37}, Pb^{204}, Pb^{206}, Pb^{207}, Pb^{208}, Cu^{63}, Cu^{65}, Zn^{64}, Zn^{66}, Zn^{67}, Zn^{68}, Zn^{70}, Ar^{36}, Ar^{38}, Ar^{40}, Si^{28}, Si^{29}, Si^{30}, B^{10}, B^{11}, Na^{23}, Fe^{54}, Fe^{56}, $Fe57$, Fe^{58}, Co^{59}.

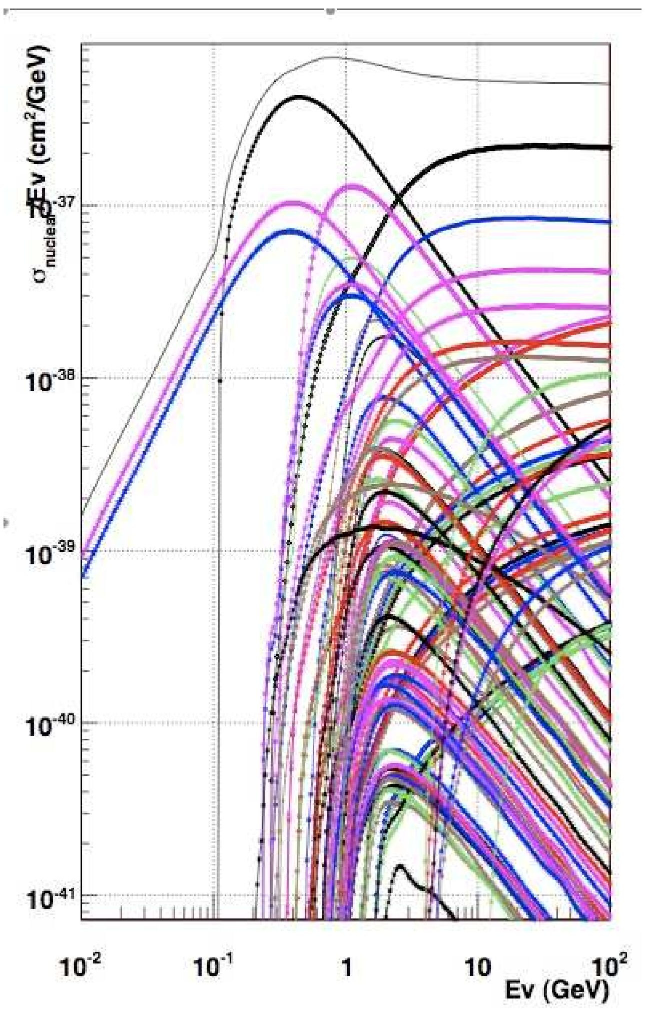

Figure 4.1: Cross section splines just for $\nu_\mu Fe^{56}$ processes modeled in GENIE. The large number of splines and the fine numerical integration stepping makes spline calculation a very CPU-intensive process.

and that you want to feed that in the calculation of splines for all ν_e, $\bar{\nu_e}$, ν_μ, $\bar{\nu_\mu}$, ν_τ, $\bar{\nu_\tau}$ scattering processes off O^{16} (PDG code: 1000080160) and Fe^{56} (PDG code: 1000260560). You need to type:

```
shell$ export GSPLOAD=/path/freenuc.xml
shell$ gmkspl -p 12,-12,14,-14,16,-16 -t 1000080160,1000260560 -o /path/spl.xml
```

The output file '*/path/spl.xml*' includes both the calculated nuclear splines and the input free nucleon splines. Feeding-in the free nucleon splines enables GENIE to finish the above calculation within a few minutes[2]. Before running a MC job using the new spline XML file you need to update the '`GSPLOAD`' environmental variable to point to that new file. The environment-driven behavior of GENIE will be explained at a following chapter.

4.2.5 Modifying GENIE and re-using splines

You should *never* be doing that (unless you are absolutely sure about what you are doing). The safest assumption is that changes in GENIE, either a change of default model parameter or a change of a default model, *invalidates* previously generated cross section splines as the cross section models (used for generating these splines) may be affected.

4.2.6 The XML cross section splines file

The XML files, you have just learnt how to generate, are particularly suited for moving data between different GENIE applications. This is the only intended usage of these files. If you wish to use GENIE's cross section splines in another context, eg. within your analysis code, then we recommend converting them from an XML to a ROOT format using GENIE's *gspl2root* utility (described later in this manual). The *gspl2root* utility can be used to write-out the cross section splines for all modeled processes into a single ROOT file, organizing them in ROOT *TDirectories* - one per initial state (eg. cross section splines for all $\nu_\mu O^{16}$ processes are stored in a 'nu_mu_O16' directory, while splines for $\bar{\nu_\mu} C^{12}$ are stored in a 'nu_mu_bar_C12' directory etc.). Each spline is stored as a ROOT *TGraph*. The *TGraph* objects can be used for numerical interpolation (using '*TGraph::Eval(double)*') so, essentially, **one can write-out all GENIE cross section 'functions' he needs into a single ROOT file** (see section on GENIE's *gspl2root* utility for more details on this particularly useful feature).

However, it is generally usefull that you do have an understanding of how the XML cross section spline file is structured. All XML splines are stored within '`<genie_xsec_spline_list>`' tags:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- generated by genie::XSecSplineList::SaveSplineList() -->
<genie_xsec_spline_list version="2.00" uselog="1">
... ... ...
... ... ...
</genie_xsec_spline_list>
```

[2]On a Intel MacBookPro Core 2 Duo processor at 2.16 GHz.

The 'uselog="1"' flag indicates that the spline knots are spaced 'logarithmically' in energy (This is the default GENIE option so that there is higher knot density where the cross section changes more rapidly). The data for each spline are stored within '`<spline>`' tags[3]:

```
<spline
    name   = "{algorithm/reaction; string}"
    nknots = "{number of knots; int}">
<knot>
  <E>    {energy; double}        </E>
  <xsec> {cross section; double} </xsec>
</knot>
<knot>
  <E>    {energy; double}        </E>
  <xsec> {cross section; double} </xsec>
</knot>
... ...
</spline>
```

Each spline is named by combining the names of the cross section algorithm and its configuration with a string interaction code. These rather long names are built automatically by GENIE and used for retrieving the correct spline[4] from the spline pool. For example, a spline named '*genie::DISPartonModelPXSec/CC-Default/nu:-12;tgt:1000260560;N:2112;q:-1(s);proc:Weak[CC],DIS*' indicates that it was computed using the cross section algorithm '*genie::DISPartonModelPXSec*' run in the '*CC-Default*' configuration for an interaction channel with the following string code: '*nu:-12;tgt:1000260560;N:2112;q:-1(s);proc:Weak[CC],DIS*' (indicating a DIS CC $\nu_\mu Fe^{56}$ scattering process of a sea $\bar{d}$ quark in a bound neutron). The spline knots are listed in increasing energy, going up to a maximum value specified during the spline construction. One of the knots falls exactly on the energy threshold for the given process so as to improve the accuracy of numerical interpolation around threshold. The energy and cross section values are given in the natural system of units ($\hbar = c = 1$) used internally within GENIE (note that the more convenient cross section units, $10^{-38}cm^2$, are used when the cross section data are exported for you using the *gspl2root* utility).

4.3 Simple event generation cases

4.3.1 The *gevgen* generic event generation driver

A number of drivers are included in GENIE for handling event generation cases of varying complexity: From trivial ones, corresponding to a fixed initial state at a fixed energy, to the most complicated ones, involving the outputs of detailed neutrino beam-line simulations and detailed Geant4-based detector geometry descriptions derived from CAD engineering drawings. The most involved cases will be described in the next chapter.

[3]In the description below, the curly braces within tags are to be 'viewed' as a single value of the specified type with the specified semantics.

[4]GENIE takes the safest route and checks both the 'reaction mode' and 'cross section algorithm'. It will not use cross section spline data calculated by a cross section algorithm A, if an alternative cross section algorithm B is currently in use.

Here, we present a simple, generic GENIE event generation driver, *gevgen*, handling all simple event generation cases. The *gevgen* driver can handle event generation for neutrinos scattered off a fixed target (or a 'target mix'). It handles mono-energetic neutrinos or neutrinos from selected from fluxes that can be described in simple terms (either via a functional form, a vector file or a ROOT *TH1D* histogram).
The source code for this utility may be found in '*$*`GENIE`*/src/stdapp/*gEvGen.cxx'.

In summary, the *gevgen* syntax is:

```
shell$ gevgen [-h] -n nev [-s] -e E -p nu -t tgt [-r run#] [-f flux] [-w]
```

where [] denotes an optional argument

Details on the command-line arguments can be found below:

- **-h** Prints-out help on *gevgen* syntax and exits.
- **-n** Specifies the number of events to generate.
- **-r** Specifies the MC run number.
- **-s** Instructs the driver to enable cross section spline interpolation.

  Always use this option as the alternative of integrating the differential cross sections at event generation time, although it is permitted, is CPU-intensive and impractical. The required cross section can be pre-computed using the *gmkspl* utility (see prev. section on 'cross section splines').

  The input cross section spline XML file is specified via the '`GSPLOAD`' environmental variable (see next section).

  If the '-s' option is specified, then the driver will check whether the full set of cross section splines, as required for the enabled processes and specified initial states, is indeed loaded. If not, then it will patch the spline list by calculating the missing splines on the fly.
- **-e** Specifies the neutrino energy or energy range.

  For example, specifying '`-e 1.5`' will instruct *gevgen* to generate events at 1.5 GeV.

  If what follows '`-e`' is a comma separated pair of values then gevgen will interpret that as an 'energy range'. For example, specifying '`-e 0.5,2.3`' will be interpreted as the [0.5 GeV, 2.3 GeV] range. If an energy range is specified then *gevgen* expects the '`-f`' option to be set as well so as to describe the energy spectrum of flux neutrinos over that range (see below).
- **-p** Specifies the neutrino PDG code.

- **-t** Specifies the target PDG code

  The PDG convention for ion codes is 10LZZZAAAI (see section on the GENIE *GHEP* event record for details).

  Multiple targets (a 'target mix') can be specified as a comma-separated list of PDG codes, each followed by its corresponding weight fraction in brackets as in:
  '`code1[fraction1],code2[fraction2],...`'.
  For example, to use a target mix of 95% O16 and 5% H type:
  '`-t 1000080160[0.95],1000010010[0.05]`'.

- **-f** Specifies the neutrino flux spectrum.

  This generic event generation driver allows to specify the flux in any one of three simple ways:

  - As a 'function'.
    For example, in order to specify a flux that has the $x^2 + 4e^{-x}$ functional form, type:
    '`-f 'x*x+4*exp(-x)`''

  - As a 'vector file'.
    The file should contain 2 columns corresponding to energy (in GeV), flux (in arbitrary units).
    For example, in order to specify that the flux is described by the vector file '*/data/fluxvec.data*', type:
    '`-f /data/fluxvec.data`'

  - As a '1-D histogram (*TH1D*) in a ROOT file'.
    The general syntax is: '`-f /full/path/file.root,object_name`'.
    For example, in order to specify that the flux is described by the 'nue' *TH1D* object in '*/data/flux.root*', type:
    '`-f /data/flux.root,nue`'

- **-w** Forces generation of weighted events.

  This option is relevant only if a neutrino flux is specified via the '`-f`' option. In this context 'weighted' refers to an event generation biasing in selecting an initial state (a flux neutrino and target pair at a given neutrino energy). Internal weighting schemes for generating event kinematics can still be enabled independently even if '-w' is not set. Don't use this option unless you understand what the internal biasing does and how to analyze the generated sample. The default option is to generated unweighted events.

Examples

1. To generate 20,000 ν_μ (PDG code: 14) scattered off Fe^{56} (PDG code: 1000260560) at an energy of 6.5 GeV, type:

```
shell$ gevgen -n 20000 -s -e 6.5 -p 14 -t 1000260560
```

The -s option will instruct the code to use pre-computed cross section (from an XML file generated using the *gmkspl* utility and specified via the '`GSPLOAD`' environmental variable - See next section).

2. To generate a similar sample as above, but with the ν_μ energies, between 1 and 4 GeV, selected from a spectrum that has the $x^2 e^{(-x^2+3)/4}$ functional form, type:

```
shell$ gevgen -n 20000 -s -e 1,4 -p 14 -t 1000260560
-f 'x*x*exp((-x*x+3)/4)'
```

3. To generate a similar sample as above, but with the neutrino flux described via the '*/path/flux.data*' input vector file, type:

```
shell$ gevgen -n 20000 -s -e 1,4 -p 14 -t 1000260560
-f /path/flux.data
```

4. To generate a similar sample as above, but with the neutrino flux described a ROOT *TH1D* histogram called 'nu_flux' stored in '*/path/file.root*', type:

```
shell$ gevgen -n 20000 -s -e 1,4 -p 14 -t 1000260560
-f /path/file.root,nu_flux
```

Note that the event generation driver will use only the input histogram bins that fall within the specified (via the '`-e`' option) energy range. In the example shown above, all the neutrino flux bins that do not fall in the 1 to 4 GeV energy range will be neglected. The bins including 1 GeV and 4 GeV will be taken into account. So the actual energy range used is: from the lower edge of the bin containing 1 GeV to the upper edge of the bin containing 4 GeV.

5. To generate a similar sample as above, but, this time, on a target mix that is made of 95% O16 (PDG code: 1000080160) and 5% H (1000010010), type:

```
shell$ gevgen -n 30000 -s -e 1,4 -p 14
-t 1000080160[0.95],1000010010[0.05] -f /path/file.root,nu_flux
```

4.3.2 Event generation outputs

During event generation a status file, containing MC job statistics and the most recent event dump, is being updated periodically (see next section of how to control its refresh rate). In default-mode a lot of information is displayed on the terminal (see next section on how to control the GENIE verbosity level on a per message stream basis). When the MC job finishes then a ROOT file named

'*<prefix>.ghep.root*' will have been written-out. It contains a *GHEP* MC event tree ('gtree') and a couple of ROOT folders, 'gconfig' and 'genv', containing, respectivelly, snapshots of your GENIE configuration and running environment. You can open the file in an interactive ROOT session and browse these ROOT folders. Chapter 5 describes how to set up an 'event loop' and analyze the generated sample.

4.4 Environment-driven GENIE behavior

In this section we describe the effects of the GENIE environmental variables. Examples given below are for the bash shell (Modify accordingly for other shells).

4.4.1 Changing the MC job seed number

You set the GENIE MC seed number via the '`GSEED`' environmental variable. For example, type:

```
shell$ export GSEED=19829839
```

before running an MC job, to set the seed number to 19829839.

4.4.2 Overridding the default list of event generation threads

The set of event generation threads loaded into the event generation drivers is controlled by the '`GEVGL`' environmental variables. If unset, GENIE will load a set of event generation threads that correspond to a comprehensive desription of event generation physics. You may wish to override that if you want to generate specific interaction modes only or if you to override the default GENIE threads. For example, in order to instruct the event generation drivers to generate only QEL events, type:

```
shell$ export GEVGL=QEL
```

before running the MC job.

Possible '`GEVGL`' values can be found at:
'`$GENIE`*/config/EventGeneratorListAssembler.xml.*'
The sets of event generation threads defined there is extensibe.

4.4.3 Overridding the default message stream thresholds

The default thresholds are defined in '`$GENIE`*/config/Messenger.xml*'. This file controls the GENIE verbosity. If you wish to override the default threshold for a particular stream, either because you want to print-out low level debug info not shown by default, or because you want to suppress messages that do get shown by default, then:

1. Create a file similar to '*Messenger.xml*' at some local directory.

2. List only the streams whose default priority you want to override and set your preferred thresholds.
3. Set the '`GMSGCONF`' environmental variable to point to that file.

4.4.4 Setting GENIE in production mode verbosity

Setting the '`GPRODMODE`' environmental variable to 'YES' will enforce a minimal verbosity level appropriate for large scale production runs: All message stream threshold levels will be raised to 'WARN' so that all 'DEBUG', 'INFO' & 'NOTICE' messages will be filtered and only 'WARN', 'ERROR', and 'FATAL' messages may be shown. To enable the production-mode verbosity just type:

```
shell$ export GPRODMODE=YES
```

To return to the default behavior, unset the '`GPRODMODE`' variable:

```
shell$ unset GPRODMODE
```

4.4.5 Override the default GENIE configuration folder

In a multiple-user environment, users may want to use a single, common GENIE installation but be able to tweak the GENIE configuration and behavior independently without affecting other users. GENIE can be instructed to load its configuration from a (non-default) user directory:

1. Copy the entire '`$GENIE`*/config*' directory to a user direcory, for example: '*/path/my_genie_config/*'.
2. Set the '`GALGCONF`' environmental variable to point to that private directory:

   ```
   shell$ export GALGCONF=/path/my_genie_config/
   ```

GENIE jobs running in an environment where '`GALGCONF`' is set will load the configuration from the user directory, rather from the default directory. GENIE jobs run by other users, in an environment where '`GALGCONF`' is unset (or set to a different value), will remain unaffected.

4.4.6 Accepting unphysical events

GENIE events include a ROOT *TBits* error flag. By default, if any bit is set during event generation then the event is discarded. Users may choose to accept certain types of unphysical events by setting the '`GUNPHYSMASK`' environmental variable, a bitfield mask instructing GENIE to ignore certain error bits (corresponding to certain types of unphysical events).

[expand]

4.4.7 Specifying input XML cross section file

The '`GSPLOAD`' environmental variable can be used to specify the XML cross section spline file to be read-in by the event generation drivers.

```
shell$ export GSPLOAD=/path/to/file.xml
```

4.4.8 Specifying output XML cross section file

The '`GSPSAVE`' environmental variable can be used to specify an XML file for writting-out the cross section splines that were used during a MC job. This is only usefull when the event generation driver determines that the input set of splines (loaded from the XML file specified via the '`GSPLOAD`' variable) was not a complete set of splines for the event generation case at hand (which indicates some sloppiness on your side) and had the missing splines calculated. Setting the '`GSPSAVE`' variable in this case will allow the extended set of splines to be written out.

4.4.9 Specifying cache file

At event generation time certain repetitive, time-consuming operations, such as finding the maximum differential cross section (for a given process over the entire phase space at a fixed energy) so as to be used in rejection method-based kinematical selections etc, are cached by GENIE (its performance improves with time during an event jeneration job). To write-out the internal cache buffers at a ROOT file so that can be fed-into subsequent jobs one can use the '`GCACHEFILE`' environmental variables. To write out the internal cache into *'/some/path/mycache.root'*, then type:

```
shell$ export GCACHEFILE=/some/path/mycache.root
```

If the '`GCACHEFILE`' variable is unset then the internal cache is deleted at the end of the MC job. Modifying GENIE can potentially invalidate previously cached data.

4.4.10 Overriding the default set of global defaults

The most important GENIE user configuration parameters are collected in the `$GENIE`*/config/UserPhysicsOptions.xml' f*ile. Different named sets of these parameters may exist in that file. By default, GENIE reads-in the 'Default' set. To force GENIE to read-in an alternative parameter set the specify its name using the `'GUSERPHYSOPT'` environmental variable. For example, in order to set a (hypothetic) T2K-preferred set of GENIE configuration parameters stored in the *'UserPhysicsOptions.xml'* under the 'T2K' name, type:

```
shell$ export GUSERPHYSOPT=T2K
```

4.4.11 Overriding the default GHEP print-out verbosity

The *GHEP* print-out verbosity can be controlled via the '`GHEPPRINTLEVEL`' environmental variable. Possible values are:

- '0' : prints-out the particle list.
- '1' : prints-out the particle list and event flags.
- '2' : prints-out the particle list and event flags and cross sections / weights.
- '3' : prints-out the particle list and event flags and cross sections / weights and event summary info.
- '10' : similar to '0' but particle positions are printed-out too.
- '11' : similar to '1' but particle positions are printed-out too.
- '12' : similar to '2' but particle positions are printed-out too.
- '13' : similar to '3' but particle positions are printed-out too.

To print-out the *GHEP* record including abridged particle list info, event flags, event cross sections / weights and event summary info, type:

```
shell$ export GHEPPRINTLEVEL=3
```

See the *GHepRecord::Print()* method for more details.
The default behavior is the one corresponding to a *'*`GHEPPRINTLEVEL`*'* value of '1'.

4.4.12 Overriding the default status file refresh rate

During event generation a status file (named *'genie-mcjob-[run number].status'*) is written out and updated periodically. The file contains the current event number, a print-out of the last generated event, the integrated processing time and the average processing time per event and is quite usefull for monitoring the event generation progress. By default, the file is refreshed every 100 events. One can modify the refresh rate by setting the *'*`GMCJMONREFRESH`*'* environmental variable. For example, in order to set the refresh rate to 5 events, type:

```
shell$ export GMCJMONREFRESH=5
```

Chapter 5

Using realistic fluxes and detector geometries

5.1 Introduction

The main task of GENIE is to simulate the complex physics processes taking place when a neutrino is scattered off a nuclear target. The generator employs advanced, heavily validated models to describe the primary scattering process, the neutrino-induced hadronic multiparticle production and the intra-nuclear hadron transport and re-scattering.

Event generation for realistic experimental setups presents neutrino generators with additional computational challenges. The physics generator is required to handle a large number of nuclear targets (ranging from as light as H^1 to as heavy as Pb^{208}). Moreover, when simulating neutrino interactions in detectors (such as the JPARC and NuMI near detectors) exposed to a non-uniform neutrino flux changing rapidly across the detector volume, it is particularly important to take into account both the detailed detector geometry and the spatial dependencies of the flux. This ensures the proper simulation of backgrounds and avoids introducing highly non-trivial MC artifacts.

The GENIE framework provides many off-the-shelf components for simulating neutrino interactions in realistic experimental setups. New components, encapsulating new neutrino fluxes or detector geometry descriptions, can be trivially added and seamlessly integrated with the GENIE neutrino interaction physics descriptions.

5.2 Components for building customized event generation applications

GENIE is being used by a host of precision-era neutrino experiments and strives to provide off-the-shelf components for generating neutrino interactions under the most realistic assumptions integrating the state-of-the-art GENIE neutrino interaction modeling with detailed flux and detector geometry descriptions.GENIE provides an event generation driver class, *GMCJDriver*, that can be used to setup complicated Monte Carlo jobs involving arbitrarily complex, realistic beam flux simulations and detector geometry descriptions. These flux descriptions are typically derived from experiment-specific beam-line simulations while the detector geometry descriptions are typically derived from CAD engineering drawings mapped into the Geant4, ROOT or GDML geometry description languages. Obviously, flux and detector geometry descriptions can take many forms, driven by experiment-specific choices. GENIE standardizes the geometry and flux driver interfaces. They define the operations that GENIE needs to perform on the geometry and flux descriptions and the information GENIE needs to extract from these in order to generate events. Concrete implementations of these interfaces are loaded into the GENIE event generation drivers, extending GENIE event generation capabilities and allow it to seamlessly integrate new geometry descriptions and beam fluxes.

5.2.1 Concrete neutrino flux drivers

In GENIE every concrete flux driver implements the *GFluxI* interface. The interface defines what neutrino flux information is needed by the event generation drivers and how that information is to be obtained. Each concrete flux driver includes methods to

- Declare the list of flux neutrinos that can be generated. This information is used for initialization purposes, in order to construct a list of all possible initial states in a given event generation run.
- Declare the maximum energy. Again this information is used for initialization purposes, in order to calculate the maximum possible interaction probability in a given event generation run. Since neutrino interaction probabilities are tiny and in order to boost the MC performance, GENIE scales all interaction probabilities in a particular event generation run so that the maximum possible interaction probability is 1. That maximum interaction probability corresponds to the total interaction probability (summed over nuclear targets and process types) for a maximum energy neutrino following a trajectory that maximizes the density-weighted path-lengths for each nuclear target in the geometry. GENIE adjusts the MC run normalization accordingly to account for that internal weighting.
- Generate a flux neutrino and specify its pdg code, its weight (if any), its 4-momentum and 4-position. The 4-position is given in the detector coordinate system (as specified by the input geometry). Each such flux neutrino is propagated towards the detector geometry but is not required to cross any detector volume. GENIE will take that neutrino through the geometry, calculate density-weighted path-lengths for all nuclear targets in the geometry, calculate the corresponding interactions probability off each nuclear target and decide whether that flux neutrino should interact. If it interacts, an appropriate *GEVGDriver* will be invoked to generate the event kinematics.
- Notify that no more flux neutrinos can be thrown. This flag is typically raised by flux drivers that simply read-in beam-line simulation outputs (as opposed to run the beam simulation code on the fly) so as to notify

GENIE that the end of the neutrino flux file has been reached (after, probably, having been recycled N times). The flag allows GENIE to properly terminate the event generation run at the end-of-flux-file irrespective of the accumulated number of events, protons on target, or other metric of exposure.

The above correspond the the common set of operations /information that GENIE expects to be able to perform / extract from all concrete flux drivers. Specialized drivers may define additional information that can be utilized in the experiment-specific event generation drivers. One typical example of this is the flux-specific pass-through information, that is information about the flux neutrino parents such as the parent meson PDG code, its 4-momentum its 4-position at the production and decay points that GENIE simply attaches to each generated event and passes-through so as to be used in later analysis stages.

The latest version of GENIE at the time of writing this article (version 2.5.1) contains a host of concrete flux drivers that allow GENIE to be used in many realistic, experiment-specific situations:

- *GJPARCNuFlux*: An interface to the JPARC neutrino beam simulation [53] used at SK, nd280, and INGRID.
- *GNuMIFlux*: An interface to the NuMI beam simulations [54]used at MINOS, NOvA, MINERvA and ArgoNEUT.
- *GBartolAtmoFlux*: A driver for the BGLR atmospheric flux by G. Barr, T.K. Gaisser, P. Lipari, S. Robbins and T. Stanev [55].
- *GFlukaAtmo3DFlux*: A driver for the FLUKA atmospheric neutrino flux by A. Ferrari, P. Sala, G. Battistoni and T. Montaruli [56].
- *GCylindTH1Flux*: A generic flux driver, describing a cylindrical neutrino flux of arbitrary 3-D direction and radius. The radial dependence of the neutrino flux is configurable (default: uniform per unit area). The flux driver may be used for describing a number of different neutrino species whose (relatively normalised) energy spectra are specified as ROOT 1-D histograms. This driver is being used whenever an energy spectrum is an adequate description of the neutrino flux.
- *GMonoEnergeticFlux*: A trivial flux driver throwing mono-energetic flux neutrinos along the +z direction. More that one neutrino species can be included, each with its own weight. The driver is being used in simulating a single initial state at a fixed energy mainly for probing, comparing and validating neutrino interaction models.

New concrete flux drivers (describing the neutrino flux from other beam-lines) can be easily developed and they can be effortlessly and seamlessly integrated with the GENIE event generation framework.

5.2.2 Concrete geometry drivers

In GENIE every concrete geometry driver implements the *GeomAnalyzerI* interface. The interface specifies what information about the input geometry is relevant to the event generation and how that information is to be obtained. Each concrete geometry driver implements methods to

- Declare the list of target nuclei that can be found in the geometry. This information is used for initialization purposes, in order to construct a list of all possible initial states in a given event generation run.
- Compute the maximum density-weighted path-lengths for each nuclear target in the geometry. Again, this is information used for initialization purposes. The computed 'worst-case' trajectory is used to calculate the maximum possible interaction probability in a particular event generation run which is being used internally to normalize all computed interaction probabilities.
- Compute density-weighted path-lengths for all nuclear targets, for a 'track' of a given 4-momentum and starting 4-position. This allows GENIE to calculate probabilities for each flux neutrino to be scattered off every nuclear target along its path through the detector geometry.
- Generate a vertex along a 'track' of a given 4-momentum and starting 4-position on a volume containing a given nuclear target. This allows GENIE to place a neutrino interaction vertex within the detector geometry once an interaction of a flux neutrino off a selected nuclear target has been generated.

GENIE currently contains two concrete geometry drivers which are sufficient for all event generation cases encountered so far:

- *ROOTGeomAnalyzer*: A geometry driver handling detector geometries specified using ROOT. As detector geometries specified using Geant4 or *GDML* can be converted into ROOT geometries, this driver is being used in all cases where a detailed detector geometry is being passed on to GENIE.
- *PointGeomAnalyzer*: A trivial geometry corresponding to a single nuclear target or a target mix (a set of nuclear targets each with its corresponding weight fraction) at a fixed position. This driver is being used to simulate only given initial states as a means for probing the neutrino interaction physics modeling or in experimental situations where the detector is being illuminated by a spatially uniform neutrino beam and where the generated interaction vertices do not have any spatial dependence and can be generated uniformly within volumes of given nuclear targets.

Concluding this section, again it is worth re-emphasizing that any new detector geometry description can be seamlessly integrated with the GENIE event generation framework by means of developing an appropriate GENIE geometry driver.

5.3 Experiment-specific event generation applications

5.3.1 The T2K event generation application

This section describes *gT2Kevgen*, a GENIE-based event generation application for T2K. It integrates the GENIE with the JNUBEAM [?] JPARC neutrino beam-line simulation and the detector geometry descriptions of nd280, 2km, INGRID and Super-K.

In summary, the *gT2Kevgen* syntax is:

```
shell$ gT2Kevgen
   -g geometry
   -f flux
   [-p POT normalization of flux file]
   [-t top volume name at geom]
   [-m max path lengths xml file]
   [-L length units at geom]
   [-D density units at geom]
   [-n num of events]
   [-c num of flux cycles]
   [-e, -E exposure in POTs]
   [-o output event file prefix]
   [-r run#]
   [-h]
```

where [] denotes an optional argument.

Details on the driver options can be found below:

-g Specifies the input detector 'geometry'.

This option can be used to specify any of:

- A ROOT file containing a ROOT/Geant4-based geometry description. This is the standard option for generating events in the nd280, 2km and INGRID detectors.

  Example:
  To use the ROOT detector geometry description stored in the '*nd280-geom.root*' file, type:
  `'-g /some/path/nd280-geom.root'`

  By default the entire input geometry will be used. Use the '`-t`' option to allow event generation only on specific geometry volumes.

- A mix of target materials, each with its corresponding weight.
  This is the standard option for generating events in the Super-K detector where the beam profile is uniform and distributing the event vertices uniformly in the detector volume is sufficient. The target mix is specified as a comma-separated list of nuclear PDG codes (in the PDG2006 convention: 10LZZZAAAI) followed by their corresponding weight fractions in brackets, as in:
  `'-t code1[fraction1],code2[fraction2],...'`

  Example 1:
  To use a target mix of 95% O^{16} and 5% H type:
  `'-g 1000080160[0.95],1000010010[0.05]'`

Example 2:
To use a target which is 100% C^{12}, type:
`‘-g 1000060120’`

-f Specifies the input ‘neutrino flux’.

This option can be used to specify any of:

- A JNUBEAM flux ntuple and the detector location. The general sytax is:
  `‘-f /path/flux_file.root,detector_loc(,neutrino_list)’`

  The detector location can be any of ‘sk’ or the near detector positions ‘nd1’,...,‘nd6’ simulated by JNUBEAM. The ntuple has to be in ROOT format and can be generated from the distributed HBOOK ntuples using ROOT’s *h2root* utility.
  The optional *neutrino_ list* is a comma separated list neutrino PDG codes. It specifies which neutrino flux species to to considered in the event generation job. If no such neutrino list is specified then, by default, GENIE will consider all neutrino species in the input flux ntuple.

  When a JNUBEAM ntuple is used for describing the neutrino flux, GENIE is able to calculate the POT exposure for the generated event sample and any one of the exposure setting methods (‘`-e`’, ‘`-E`’, ‘`-c`’, ‘`-n`’, see below) can be used.

  All JNUBEAM information on the flux neutrino parent (parent PDG code, parent 4-position and 4-momentum at the production and decay points etc) are stored in a ‘flux’ branch of the output event tree and is associated with the corresponding generated neutrino event.

  Example 1:
  To use the Super-K JNUBEAM flux ntuple from the ‘*/t2k/flux/jnubeam001.root*’ file, type:
  `‘-f /t2k/flux/jnubeam001.root,sk’`

  Example 2:
  To use the 2km flux ntuple [near detector position ‘nd1’ in the jnubeam flux simulation] from the ‘*/t2k/flux/jnubeam001.root*’ file, type:
  `‘-f /t2k/flux/jnubeam001.root,nd1’`

  Example 3:
  To use the nd280 flux ntuple [near detector position ‘nd5’ in the jnubeam flux simulation] from the ‘*/t2k/flux/jnubeam001.root*’ file, type:
  `‘-f /t2k/flux/jnubeam001.root,nd5’`

  Example 4:
  To the same as above but using only the ν_e and $\bar{\nu_e}$ flux ntuple entries, type:
  `‘-f /t2k/flux/jnubeam001.root,nd5,12,-12’`

- A set of flux histograms stored in a ROOT file. The general syntax is:
  `'-f /path/file.root,neutrino_code[histo],...'`

  where *neutrino_ code* is a standard neutrino PDG code[1] and *histo* is the corresponding ROOT histogram name.

  Multiple flux histograms can be specified for different flux neutrino species (see the example given below). The relative flux normalization for all neutrino species should be represented correctly at the input histogram normalization. The absolute flux normalization is not relevant: Unlike when using JNUBEAM ntuples to describe the flux, no POT calculations are performed when plain histogram-based flux descriptions are employed. One can only control the MC run exposure via the number of generated events ('`-n`', see below). In this case the POT normalization of the generated sample is calculated externally.

  Since there is no directional information in histogram-based descriptions of the flux, the generated neutrino vertex is always set to (0,0,0,0). Then it is the detector MC responsibility to rotate the interaction vectors and plant the vertex [2] Obviously no flux pass-through branch is written out in the neutrino event tree since no such information is associated with flux neutrinos selected from plain histograms.

  Example:
  To use the histogram 'h1' (representing the ν_μ flux) and the histogram 'h2' (representing the ν_e flux) from the *'/data/flux.root'* file, type:
  `'-f /data/flux.root,14[h1],12[h2]'`

-t Specifies the input top volume for event generation.

This is an optional argument. By default, it is set to be the 'master volume' of the input geometry resulting in neutrino events being generated over the entire geometry volume. If the '`-t`' option is set, event generation will be confined in the specified detector volume. The option can be used to simulate events at specific sub-detectors.

Example:
To generate events in the P0D only, type:
`-t P0D`

Caveat:
The option was not supported in GENIE v2.4.0 used at MDC0 due to the fact that coordinate transformations between the master coordinate system and the local coordinate systems of arbitrary geometry volumes were not implemented in the GENIE geometry driver. Events at MDC0 were generated using the entire input geometry volume. The -t option has been implemented at GENIE v2.5.1 and can be used at future physics MC productions.

[1] ν_e: 12, ν_μ: 14, ν_τ: 16, $\bar{\nu}_e$: -12, $\bar{\nu}_\mu$: -14 and $\bar{\nu}_\tau$: -16

[2] This option is used only for the Super-K simulation where vertices are distributed uniformly in volume by the detector MC (SKDETSIM). For event generation at the more complex near detectors a JNUBEAM ntuple-based flux description should be used so as the interaction vertex is properly planted within the input geometry by GENIE.

-m Specifies an XML file with the maximum density-weighted path-lengths for each nuclear target in the input geometry.

This is an optional argument. If the option is not set then, at the MC job initialization, GENIE will scan the input geometry to determine the maximum density-weighted path-lengths for all nuclear targets. The computed information is used for calculating the neutrino interaction probability scale to be used in the MC job (the tiny neutrino interaction probabilities get normalized to a probability scale which is defined as the maximum possible total interaction probability, corresponding to a maximum energy neutrino in a worst-case trajectory maximizing its density-weighted path-length, summed up over all possible nuclear targets). That probability scale is also used to calculate the absolute, POT normalization of a generated event sample from the POT normalization of the input JNUBEAM flux ntuple.

Feeding-in pre-computed maximum density-weighted path-lengths results in faster MC job initialization and ensures that the same interaction probability scale is used across all MC jobs in a physics production job (the geometry is scanned by a MC ray-tracing method and the calculated safe maximum density-weighted path-lengths may differ between MC jobs).

The maximum density-weighted path-lengths for a Geant4/ROOT-based detector geometry can be pre-computed using GENIE's *gmxpl* utility.

-L Specifies the input geometry length units.

This is an optional argument.
By default, that option is set to 'mm', the length units used for the nd280 detector geometry description.
Possible options include: 'm', 'cm', 'mm', ...

-D Specifies the input geometry density units.

This is an optional argument.
By default, that option is set to 'clhep_def_density_unit', the density unit used for the nd280 detector geometry description (= ~1.6E-19 x g/cm3 !).
Possible options include: 'kg_m3', 'g_cm3', 'clhep_def_density_unit',...

-p Specifies to POT normalization of the input flux file.

This is an optional argument. By default, it is set to the standard JNUBEAM flux ntuple normalization of 1E+21 POT/detector (for the near detectors) or 1E+21 POT/cm2 (for the far detector)
That will be used to interpret the flux weights and calculate the POT normalization for the generated neutrino event sample. The option is irrelevant if a simple, histogram-based description of the neutrino flux is used (see -f option)

-c Specifies how many times to cycle a jnubeam flux ntuple.

This option provides a way to set the MC job exposure in terms of complete JNUBEAM flux ntuple cycles. On each cycle, every flux neutrino in the ntuple will be thrown towards the detector geometry.

-e Specifies how many POTs to generate.

If this option is set, *gT2Kevgen* will work out how many times it has to cycle through the input flux ntuple in order to accumulate the requested statistics. The program will stop at the earliest complete flux ntuple cycle after accumulating the required statistics. The generated statistics will slightly overshoot the requested number but the calculated exposure (which is also stored at the output file) will be exact. This option is only available with JNUBEAM ntuple-based flux descriptions.

-E Specifies how many POTs to generate.

This option is similar to '`-e`' but the program will stop immediately after the requested POT has been accumulated, without waiting for the current loop over the flux ntuple entries to be completed. The generated POT overshoot (with respect to the requested POT) will be negligible, but the POT calculation within a flux ntuple cycle is only approximate. This reflects the details of the JNUBEAM beam-line simulation. This option is only available with JNUBEAM ntuple-based flux descriptions.

-n Specifies how many events to generate.

Note that out of the 4 possible ways of setting the exposure ('`-c`', '`-e`', '`-E`', '`-n`') this is the only available one if a plain histogram-based flux description is used.

-o Sets the prefix of the output event file.

This is an optional argument. It allows you to override the output event file prefix. In GENIE, the output filename is built as:

```
prefix.run_number.event_tree_format.file_format
```

where, in *gT2Kevgen*, by default, `prefix`: 'gntp' and `event_tree_format`: 'ghep' and `file_format`: 'root'.

-r Specifies the MC run number.

This is an optional argument. By default a run number of '1000' is used.

-h Prints out the gT2Kevgen syntax and exits.

Examples

1. Generate events (run number '1001') using the jnubeam flux ntuple in '*/data/t2k/flux/07a/jnb001.root*' and picking up the flux entries for the detector location 'nd5' (which corresponds to the 'nd280m' location). The job will load the nd280 geometry from '*/data/t2k/geom/nd280.root*' and interpret it assuming the length unit is 'mm' and the density unit is the default CLHEP one. The job will stop on the first complete flux ntuple cycle after generating 5E+17 POT.

```
shell$ gT2Kevgen -r 1001 -f /data/t2k/flux/07a/jnb001.root,nd5
```

```
-g /data/t2k/geom/nd280.root -L mm -D clhep_def_density_unit -e 5E+17
```

2. As before, but now the job will stop after 100 flux ntuple cycles, whatever POT and number of events that may correspond to.

   ```
   shell$ gT2Kevgen -r 1001 -f /data/t2k/flux/07a/jnb001.root,nd5
   -g /data/t2k/geom/nd280.root -L mm -D clhep_def_density_unit -c 100
   ```

3. As before, but now the job will stop after generating 100000 events, whatever POT and number of flux ntuple cycles that may correspond to.

   ```
   shell$ gT2Kevgen -r 1001 -f /data/t2k/flux/07a/jnb001.root,nd5
   -g /data/t2k/geom/nd280.root -L mm -D clhep_def_density_unit -n 100000
   ```

4. Generate events (job number '1001') using the jnubeam flux ntuple in '*/data/t2k/flux/07a/jnb001.root*' and picking up the flux entries for the Super-K detector location. This time, the job will not use any detailed detector geometry description but just (95% O^{16} + 5% H) target-mix. The job will stop after generating 50000 events.

   ```
   shell$ gT2Kevgen -r 1001 -f /data/t2k/flux/07a/jnb001.root,sk
   -g 1000080160[0.95],1000010010[0.05] -n 50000
   ```

5. As before, but now the flux is not described using a JNUBEAM ntuple but a set of 1-D histograms from the '*/data/flx.root*' file: The histogram named 'h1' will be used for the ν_e flux, 'h2' will will be used for the $\bar{\nu_e}$ flux, and 'h3' for the ν_μ flux.

   ```
   shell$ gT2Kevgen -r 1001 -f /data/flx.root,12[h1],-12[h2],14[h3]
   -g 1000080160[0.95],1000010010[0.05] -n 50000
   ```

5.3.2 The NuMI event generation application

This section describes *g?evgen*, a GENIE-based event generation application for experiments in the NuMI beam-line. It integrates the GENIE neutrino interaction simulations with the GNUMI-based [?] beam-line simulation and the geometry descriptions of MINOS Far/Near, NoVA Far/Near, MINERvA, ArgoNEUT, MicroBooNE and other detectors.

In summary, the *g?evgen* syntax is:

[to be written]

5.4 Adding a new experiment-specific event generation application

[to be written]

Chapter 6

Analyzing GENIE outputs

6.1 Introduction

[to be written]

6.2 Understanding the GHEP event structure

6.2.1 Particles

The basic output unit of the event generation process is a 'particle'. This is an overloaded term used to describe both particles and nuclei appearing in the initial, intermediate or final state, or generator-specific pseudo-particles used for facilitating book-keeping of the generator actions.

Each such 'particle' generated by GENIE is an instance of the *GHepParticle* class. These objects contain information with particle-scope such as its particle and status codes, its pdg mass, charge and name, the indices of its mother and daughter particles marking possible associations with other particles in the same event, its 4-momentum and 4-position (in the target nucleus coordinate system), its polarization vector, and other properties. The GHepParticle class includes methods for setting and querying these properties.

GENIE has adopted the standard PDG particle codes. For ions it has adopted a PDG extension, using the 10-digit code 10LZZZAAAI where AAA is the total baryon number, ZZZ is the total charge, L is the number of strange quarks and I is the isomer number (I=0 corresponds to the ground state). GENIE-specific pseudo-particles have PDG code >= 2000000000 and can convey important information about the underlying physics model. Pseudo-particles generated by other specialized MCs that may be called by GENIE (such as PYTHIA) are allowed to retain the codes specified in that MC.

GENIE marks each particle with a status code. This signifies the position of an particle in an event and helps navigating within the event record. Most generated particles are typically marked as one of the following:

- ***'initial state'*** typically the first two particles of the event record corresponding to the incoming neutrino and the nuclear target.

- ***'nucleon target'***, corresponding to the hit nucleon (if any) within the nuclear target.
- ***'intermediate state'***, typically referring to the remnant nucleus, fragmentation intermediates such as quarks, diquarks, some intermediate pseudo-particles etc.
- ***'hadron in the nucleus'***, referring to a particle of the primary hadronic system, that is the particles emerging from the primary interaction vertex before their possible re-interactions during their intranuclear hadron transport.
- ***'decayed state'***, such as for example unstable particles that have been decayed.
- ***'stable final state'*** for the relatively long-lived particles emerging from the nuclear targets

All particles generated by GENIE during the simulation of a single neutrino interaction are stored in a dynamic container representing an 'event'.

6.2.2 Events

Events generated by GENIE are stored in a custom, *STDHEP*-like, event record called *GHEP* record. Each *GHEP* event record, an instance of the *GHepRecord* class, is a ROOT *TClonesArray* container of *GHepParticle* objects representing individual particles.

Other than being a container for the generated particles, the event record holds additional information with event-, rather than particle-, scope such as the cross sections for the selected event and the differential cross section for the selected event kinematics, the event weight, a series of customizable event flags and an interaction summary (see next section).

Additionally, the event record includes a host of methods for querying / setting event properties including many methods that allow querying for specific particles within the event (such as for example methods to return the target nucleus, the final state primary lepton or a list of all the stable descendants of any intermediate particle).

The event record features a 'spontaneous re-arrangement' feature which maintains the compactness of the daughter lists at any given time. This is necessary for the correct interpretation of the stored particle associations as the daughter indices correspond to a contiguous range. The particle mother and daughter indices for all particles in the event record are automatically updated as a result of any such spontaneous particle rearrangement.

The event generation itself is built around the *GHEP* event record using the Visitor design pattern. That interaction between the record and the event generation code will be outlined in the following sections.

The *GHEP* structure is highly compatible with the event structures used in most HEP generators. That allows us to call other generators (such as for example PYTHIA / JETSET) as part of an event generation chain and convert / append their output into the current *GHEP* event. Additionally the *GHEP* events can be converted to many other formats for facilitating the GENIE interface with experiment-specific offline software systems and cross-generator comparisons.

6.2.3 Interactions

All particles generated by GENIE for each simulated event are stored into a *GHEP* record which represents the most complete description of a generated event. Certain external heavy-weight applications such as specialized event-reweighing schemes or realistic, experiment-level MC simulation chains using the generator as the physics front-end require that detailed particle-level information.

However, many of the actual physics models employed by the generator, such as cross section, form factor or structure function models, require a much simpler event description. An event description based on simple summary information, typically including a description of the initial state, the process type and the scattering kinematics, is sufficient for driving the algorithmic objects implementing these physics models. In the interest of decoupling the physics models from event generation and the particle-level event description, GENIE uses an *Interaction* object to store summary event information. Whenever possible, algorithmic objects implementing physics models accept a single *Interaction* object as their sole source of information about an event. That enables the use of these models both within the event generation framework but also within a host of external applications such as model validation frameworks, event re-weighting tools and user physics analysis code.

An *Interaction* objects is an aggregate, hierarchical structure, containing many specialised objects holding information for the initial state (*InitialState* object), the event kinematics (*Kinematics* object), the process type (*ProcessInfo* object) and potential additional information for tagging exclusive channels (*XclsTag* object).

Users can easily instantiate *Interaction* objects and use them to drive physics models. Creating this aggregate hierarchical structure is streamlined using the 'named constructor' C++ idiom. For example, in order to define a 5 GeV QELCC $\nu_\mu + neutron$ interaction, where the neutron is bound in a Fe^{56} nucleus, (ν_μ PDG code = 14, $neutron$ PDG code = 2112, Fe^{56} PDG code = 1000260560), one needs to instantiate an *Interaction* object as in:

```
Interaction * qelcc = Interactions::QELCC(1000260560, 2112, 14, 5.0);
```

That interaction definition can be used as is to drive a QELCC cross section algorithm.

The *Interaction* objects can serialize themselves as a unique string codes which, within the GENIE framework, play the role of the 'reaction codes' of the old procedural systems. These string codes are used extensively whenever there is a need to map information to or from interaction types (as for example, mapping interaction types to pre-computed cross section splines, or mapping interaction types to specialized event generation code)

Each generated event has an *Interaction* summary object attached to it and written out in the output event trees. Despite the implications of having a certain amount of redundancy (in the sense that this summary information can be recreated entirely from the information at the *GHEP* record) this strategy presents many advantages during both event generation and analysis of generated events.

6.3 Understanding mother / daughter particle associations

[to be written]

Idx	**Name**	**ISt**	**PDG**	**Mom**	**Kids**	**E**	**px**	**py**	...
0	nu_mu	0	14	-1	4 4	...	...	...	...
1	Fe56	0	1000260560	-1	2 3				
2	neutron	11	2112	1	5 7				
3	Fe55	2	1000260550	1	10 10				
4	mu-	1	13	0	-1 -1				
5	HadrSyst	12	2000000001	2	-1 -1				
6	proton	14	211	2	-1 -1				
7	pi0	14	111	2	8 9				
8	proton	1	22	7	-1 -1				
9	pi-	1	-211	7	-1 -1				
10	**HadrBlob**	15	2000000002	3	-1 -1				

Table 6.1: [to be written - explain what is going on in this event]

Idx	**Name**	**ISt**	**PDG**	**Mom**	**Kids**	**E**	**px**	**py**	...
0	nu_mu	0	14	-1	5 5	...	...	...	...
1	Fe56	0	1000260560	-1	2 3				
2	proton	11	2212	1	4 4				
3	Mn55	2	1000250550	1	12 12				
4	Delta++	3	2224	2	6 7				
5	mu-	1	13	0	-1 -1				
6	proton	14	2112	4	8 8				
7	pi+	14	211	4	11 11				
8	proton	3	2212	6	9 10				
9	proton	1	2212	8	-1 -1				
10	proton	1	2212	8	-1 -1				
11	pi+	1	211	7	-1 -1				
12	HadrBlob	15	2000000002	3	-1 -1				

Table 6.2: [to be written - explain what is going on in this event]

Idx	**Name**	**ISt**	**PDG**	**Mom**	**Kids**	**E**	**px**	**py**	**...**
0	nu_mu	0	14	-1	4 4	...	...	...	...
1	Fe56	0	1000260560	-1	2 3				
2	neutron	11	2112	1	5 5				
3	Fe55	2	1000260550	1	22 22				
4	mu	1	13	0	-1 -1				
5	HadrSyst	12	2000000001	2	6 7				
6	u	12	2	5	8 8				
7	ud_1	12	2103	5	-1 -1				
8	string	12	92	6	9 11				
9	pi0	14	111	8	14 14				
10	proton	14	2212	8	15 15				
11	omega	12	223	8	12 13				
12	pi-	14	-211	11	16 16				
13	pi+	14	211	11	21 21				
14	pi0	1	111	9	-1 -1				
15	proton	1	2212	10	-1 -1				
16	pi-	3	-211	12	17 20				
17	neutron	1	2112	16	-1 -1				
18	neutron	1	2112	16	-1 -1				
19	proton	1	2212	16	-1 -1				
20	proton	1	2212	16	-1 -1				
21	pi+	1	211	13	-1 -1				
22	HadrBlob	15	2000000002	3	-1 -1				

Table 6.3: to be written - explain what is going on in this event

6.4 The event loop

An'event loop' skeleton is given below. You may insert your event analysis code where is indicated below. Please look at the next section for information on how to extract information from the 'event' object.

```
{
  ...
  // Open the GHEP/ROOT file
  string filename = /data/sample.ghep.root;
  TFile file(filename.c_str(), READ);

  // Get the tree header & print it
  NtpMCTreeHeader * header =
    dynamic_cast<NtpMCTreeHeader*> (file.Get("header"));
  LOG(test, pINFO) << *header;

  // Get the GENIE GHEP tree and set its branch address
  TTree * tree = dynamic_cast<TTree*> (file.Get(gtree));
  NtpMCEventRecord * mcrec = 0;
  tree->SetBranchAddress(gmrec, &mcrec);

  // Event loop
  for(Long64_t i=0; i<tree->GetEntries(); i++){
    tree->GetEntry(i);

    // print-out the event
    EventRecord & event = *(mcrec->event);
    LOG(test, pINFO) << event;

    // put your event analysis code here
    ...
    ...


    mcrec->Clear();
  }
  ...
}
```

An'event loop' skeleton can be found in *'$GENIE/src/test/testEveltLoop.cxx'*. Copy this file and use it as a starting point for your event loop.

6.5 Extracting event information

The readers are instructed to spend some time browsing the GENIE doxygen documentation, especially the classes defined in the *Interaction* and *GHEP* packages, and familiarize themselves with the public methods. Some examples on how to extract information from an 'event' objects are given below.

Examples

1. Extract the interaction summary for the given event and check whether it is a QEL CC event (excluding QEL CC charm production):

```
{
 ...

 Interaction * in = event.Summary();

 const ProcessInfo & proc = in->ProcInfo();
 const XclsTag & xclsv    = in->ExclTag();

 bool qelcc = proc.IsQuasiElastic() && proc.IsWeakCC();
 bool charm = xclsv.IsCharm();

 if (qelcc && !charm)
 {
    ...
 }
 ...
}
```

2. Get the energy threshold for the given event:

```
{
 ...

 Interaction * in = event.Summary();

 double Ethr = in->PhaseSpace().Threshold();
 ...
}
```

3. Get the momentum transfer Q^2 and hadronic invariant mass W, as generated during kinematical selection, for RES CC event:

```
{
 ...
 const ProcessInfo & proc  = in->ProcInfo();
 const Kinematics  & kine = in->Kine();
 bool selected = true;

 if (proc.IsResonant() && proc.IsWeakCC())
 {
     double Q2s = kine.Q2(selected);
     double Ws  = kine.W (selected);
 }
 ...
}
```

4. Calculate the momentum transfer Q^2, the energy transfer ν, the Bjorken x variable, the inelasticity y and the hadronic invariant mass Wdirectly from the event record:

```
{
  ...
  // get the neutrino, f/s primary lepton and hit
  // nucleon event record entries
  //
  GHepParticle * neu = event.Probe();
  GHepParticle * fsl = event.FinalStatePrimaryLepton();
  GHepParticle * nuc = event.HitNucleon();

  // the hit nucleon may not be defined
  // (eg. for coherent, or ve- events)
  //
  if(!nuc) return;

  // get their corresponding 4-momenta (@ LAB)
  //
  const TLorentzVector & k1 = *(neu->P4());
  const TLorentzVector & k2 = *(fsl->P4());
  const TLorentzVector & p1 = *(nuc->P4());

  // calculate the kinematic variables
  // (eg see Part.Phys. booklet, page 191)
  //
  double M = kNucleonMass;

  TLorentzVector q = k1 - k2;

  double Q2 = -1 * q.M2();
  double v  = q.Energy();
  double x  = Q2 / (2*M*v);
  double y  = v / k1.Energy();
  double W2 = M*M - 2*M*v - Q2;
  double W  = TMath::Sqrt(TMath::Max(0., W2));
  ...
}
```

5. Loop over particles and count the number of final state pions:

```
{
 ...
 int npi = 0;

 TObjArrayIter iter(&event);
 GHepParticle * p = 0;

 // loop over event particles
 for((p = dynamic_cast<GHepParticle *>(iter.Next()))) {

    int pdgc   = p->Pdg();
    int status = p->Status();

    if(status != kIStStableFinalState) continue;

    bool is_pi = (pdgc == kPdgPiP ||
                  pdgc == kPdgPi0 ||
                  pdgc == kPdgPiM);

    if(is_pi) npi++;
 }

 ...
}
```

6. Get the corresponding NEUT reaction code for a GENIE event:

```
{
  ...

  int neut_code = utils::ghep::NeutReactionCode(&event);
  ...
}
```

6.6 Event tree conversions

You do not need to convert the GENIE *GHEP* trees in order to analyze the generated samples or pass them on to a detector-level Monte Carlo. But you can do so if:

- you need to pass GENIE events to legacy systems using already standardized formats,
- you want to be able to read-in GENIE events without loading any GENIE libraries (eg bare-ROOT, or XML formats),
- you want to extract just summary information and write it out in simpler ntuples.

6.6.1 The *gtpc* ntuple conversion utility

The *gntpc* utility converts the native GENIE *GHEP* event tree file to a host of plain text, XML or bare-ROOT formats. More output formats can be added easily. The *gntpc* source code can be found in *'$GENIE/src/stdapp/gNtpConv.cxx'* .

In summary, the *gntpc* syntax is:

```
shell$ gntpc -i input_file [-o output_file] -f format [-n nev]
```

where [] denotes an optional argument.

Details on the utility options can be found below:

- **-n** Specifies the number of events to convert.

  By default, gntpc will convert all events in the input file.

- **-i** Specifies an input GENIE *GHEP* event tree file.

- **-f** Specifies the output file format.

  This can be any of the following strings:
  'gst', 'gxml', 'rootracker', 't2k_rootracker', ''numi_rootracker, 't2k_tracker', or 'nuance_tracker'.
  Details on these formats are given below.

  Additional formats are supported by *gntpc* but they are used mainly for GENIE testing purposes and will not be described here.

- **-o** Specifies the output file name.

  By default, if no argument is specified via the '-o' option, the output file name is constructed from the input *GHEP* file name by removing the '.ghep.root' (or '.root') extension and by appending:

  - 'gst' format: '.gst.root'
  - 'gxml' format: '.gxml'
  - 'rootracker' format: '.gtrac.root'

  - ‘t2k_rootracker’ format: ‘.gtrac.root’
  - ‘numi_rootracker’ format: ‘.gtrac.root’
  - ‘t2k_tracker’ format: ‘.gtrac.dat’
  - ‘nuance_tracker’ format: ‘.gtrac_legacy.dat’

Examples:

1. To convert all events in the *input GHEP* file ‘*myfile.ghep.root*’ into the ‘t2k_rootracker’ format, type:

   ```
   shell$ gntpc -i myfile.ghep.root -f t2k_rootracker
   ```

   The output file is named automatically ‘*myfile.gtrac.root*’

2. To convert the first 20,000 events in the GHEP file ‘*myfile.ghep.root*’ into the ‘gst’ format and name the output file ‘*out.root*’, type:

   ```
   shell$ gntpc -i myfile.ghep.root -f gst -n 20000 -o out.root
   ```

6.6.2 Formats supported by *gntpc*

6.6.2.1 The ‘gst’ format

The ‘gst’ is a GENIE summary ntuple format. It is a simple, plain ntuple that can be easily used for plotting in interactive ROOT sessions. The stored ROOT *TTree* contains the following branches:

- **iev** (*int*): Event number.
- **neu** (*int*): Neutrino PDG code.
- **tgt** (*int*): Nuclear target PDG code (10LZZZAAAI).
- **Z** (*int*): Nuclear target Z.
- **A** (*int*): Nuclear target A.
- **hitnuc** (*int*): Hit nucleon PDG code (not set for coherent, inverse muon decay and ve- elastic events).
- **hitqrk** (*int*): Hit quark PDG code (set for deep-inelastic scattering events only).
- **sea** (*bool*): Hit quark is from sea (set for deep-inelastic scattering events only).
- **resid** (*bool*): Produced baryon resonance id (set for resonance events only).
- **qel** (*bool*): Is it a quasi-elastic scattering event?
- **res** (*bool*): Is it a resonanec neutrino-production event?
- **dis** (*bool*): Is it a deep-inelastic scattering event?

- **coh** (*bool*): Is it a coherent meson production event?
- **dfr** (*bool*): Is it a diffractive meson production event?
- **imd** (*bool*): Is it an invese muon decay event?
- **nuel** (*bool*): Is it a ve- elastic event?
- **cc** (*bool*): Is it a CC event?
- **nc** (*bool*): Is it a NC event?
- **charm** (*bool*): Produces charm?
- **neut_code** (*int*): The equivalent NEUT reaction code (if any).
- **nuance_code** (*int*): The equivalent NUANCE reaction code (if any).
- **wght** (*double*): Event weight.
- **xs** (*double*): Bjorken x (as was generated during the kinematical selection / off-shell kinematics).
- **ys** (*double*): Inelasticity y (as was generated during the kinematical selection / off-shell kinematics).
- **ts** (*double*): Energy transfer to nucleus (nucleon) at coherent (diffractive) production events (as was generated during the kinematical selection).
- **Q2s** (*double*): Momentum transfer Q^2 (as was generated during the kinematical selection / off-shell kinematics) (in GeV^2).
- **Ws** (*double*): Hadronic invariant mass W (as was generated during the kinematical selection / off-shell kinematics).
- **x** (*double*): Bjorken x (as computed from the event record).
- **y** (*double*): Inelasticity y (as computed from the event record).
- **t** (*double*): Energy transfer to nucleus (nucleon) at coherent (diffractive) production events (as computed from the event record).
- **Q2** (*double*): Momentum transfer Q^2 (as computed from the event record) (in GeV^2).
- **W** (*double*): Hadronic invariant mass W (as computed from the event record).
- **Ev** (*double*): Incoming neutrino energy (in GeV).
- **pxv** (*double*): Incoming neutrino px (in GeV).
- **pyv** (*double*): Incoming neutrino py (in GeV).
- **pzv** (*double*): Incoming neutrino pz (in GeV).
- **En** (*double*): Initial state hit nucleon energy (in GeV).
- **pxn** (*double*): Initial state hit nucleon px (in GeV).

- **pyn** (*double*): Initial state hit nucleon py (in GeV).
- **pzn** (*double*): Initial state hit nucleon pz (in GeV).
- **El** (*double*): Final state primary lepton energy (in GeV).
- **pxl** (*double*): Final state primary lepton px (in GeV).
- **pyl** (*double*): Final state primary lepton py (in GeV).
- **pzl** (*double*): Final state primary lepton pz (in GeV).
- **nfp** (*int*): Number of final state p and $\bar{p}$ (after intranuclear rescattering).
- **nfn** (*int*): Number of final state n and $\bar{n}$.
- **nfpip** (*int*): Number of final state π^+.
- **nfpim** (*int*): Number of final state π^-.
- **nfpi0** (*int*): Number of final state π^0.
- **nfkp** (*int*): Number of final state K^+.
- **nfkm** (*int*): Number of final state K^-.
- **nfk0** (*int*): Number of final state K^0 and $\bar{K^0}$.
- **nfem** (*int*): Number of final state γ, $e^-$and e^+.
- **nfother** (*int*): Number of heavier final state hadrons (D+/-,D0,Ds+/-,Lamda,Sigma,Lamda_c,Sigma_c,...).
- **nip** (*int*): Number of 'primary' ('primary' : before intranuclear rescattering) p and $\bar{p}$**.**
- **nin** (int): Number of 'primary' n and $\bar{n}$.
- **nipip** (*int*): Number of 'primary' π^+.
- **nipim** (*int*): Number of 'primary' π^-.
- **nipi0** (*int*): Number of 'primary' π^0.
- **nikp** (*int*): Number of 'primary' K^+.
- **nikm** (*int*): Number of 'primary' K^-.
- **nik0** *(int)*: Number of 'primary' K^0 and $\bar{K^0}$.
- **niem** (*int*): Number of 'primary' γ, $e^-$and e^+ (eg from nuclear de-excitations or from pre-intranuked resonance decays).
- **niother** (*int*): Number of other 'primary' hadron shower particles.
- **nf** (*int*): Number of final state particles in hadronic system.
- **pdgf** (*int[kNPmax]*): PDG code of k^{th} final state particle in hadronic system.

- **Ef** (*double[kNPmax]*): Energy of k^{th} final state particle in hadronic system (in GeV).
- **pxf** (*double[kNPmax]*): Px of k^{th} final state particle in hadronic system (in GeV).
- **pyf** (*double[kNPmax]*): Py of k^{th} final state particle in hadronic system (in GeV).
- **pzf** (*double[kNPmax]*): Pz of k^{th} final state particle in hadronic system (in GeV).
- **ni** (*int*): Number of particles in the 'primary' hadronic system ('primary' : before intranuclear rescattering).
- **pdgi** (*int[kNPmax]*): PDG code of k^{th} particle in 'primary' hadronic system.
- **Ei** (*double[kNPmax]*): Energy of k^{th} particle in 'primary' hadronic system (in GeV).
- **pxi** (*double[kNPmax]*): Px of k^{th} particle in 'primary' hadronic system (in GeV).
- **pyi** (*double[kNPmax]*): Py of k^{th} particle in 'primary' hadronic system (in GeV).
- **pzi** (*double[kNPmax]*): Pz of k^{th} particle in 'primary' hadronic system (in GeV).
- **vtxx** (*double*): Vertex x in detector coord system (in SI units).
- **vtxy** (*double*): Vertex y in detector coord system (in SI units).
- **vtxx** (*double*): Vertex z in detector coord system (in SI units).
- **vtxt** (*double*): Vertex t in detector coord system (in SI units).
- **calresp0** (*double*): An approximate calorimetric response to the generated hadronic vertex actibity, calculated by summing up: the kinetic energy for generated $\{\pi^+, \pi^-, p, n\}$, the energy+mass for generated $\{\bar{p}, \bar{n}\}$, the (e/h)*energy for generated $\{\pi^0, \gamma, e^-, e^+\}$ (with an e/h = 1.3) and the kinetic energy for any other generated particle.

Examples

Examples on how to extract / plot information from the 'gst' summary ntuples are given below

1. To draw a histogram of the final state primary lepton energy for all ν_μ CC DIS interactions with an invariant mass $W > 3$ GeV, then type:
```
root[0] gst->Draw("El","dis&&cc&&neu==14&&Ws>3");
```
2. To draw a histogram of all final state π^+ energies in CC RES interactions, then type:
```
root[0] gst->Draw("Ef","pdgf==211&&res&&cc");
```

6.6.2.2 The 'gxml' format

The 'gxml' format is a GENIE XML-based event format[1].

Each event is included within <ghep> </ghep> tags as in:

```
<ghep np         = "{number of particles; int}"
      unphysical = "{is it physical?; boolean (T/F)}">

</ghep>
```

Both information with event-wide scope such as:

```
<wght>      {event weight; double}        </wght>
<xsec_evnt> {event cross section; double} </xsec_evnt>
<xsec_kine> {cross section for event kinematics; double} </xsec_kine>

<vx> {vertex x in detector coord system (SI); double} </vx>
<vy> {vertex y in detector coord system (SI); double} </vy>
<vz> {vertex z in detector coord system (SI); double} </vz>
<vt> {vertex t (SI); double} </vt>
```

and a full list of the generated particles is included between the <ghep> tags. The information for each generated particle is expressed as:

```
<p idx  = "{particle index in event record; int}"
   type = "{particle type; char (F[ake]/P[article]/N[uleus])}">

   <pdg> {pdg code; int}    </pdg>
   <ist> {status code; int} </ist>

   <mother>
            <fst> {first mother index; int} </fst>
            <lst> {last  mother index; int} </lst>
   </mother>
   <daughter>
            <fst> {first daughter index; int} </fst>
            <lst> {last  daughter index; int} </lst>
   </daughter>

   <px> {Px in GeV; double}  </px>
   <py> {Py in GeV; double} </py>
   <pz> {Pz in GeV; double}  </pz>
   <E>  {E  in GeV; double}  </E>
   <x>  {x  in fm;  double}  </x>
```

[1]In the format description that follows, the curly braces within tags are to be 'viewed' as a single value of the specified type with the specified semantics. For example '{number of particles; int}' is to be thought of as an integer value describing a number particles.

```
   <y>  {y  in fm;  double}  </y>
   <z>  {z  in fm;  double}  </z>
   <t>  {t; always set to 0} </t>

   <ppolar> {polarization, polar angle;     in rad} </ppolar>
   <pazmth> {polarization, azimuthal angle; in rad} </pazmth>
</p>
```

6.6.2.3 The 'rootracker' formats

The 'rootracker' format is a standardized bare-ROOT GENIE event tree format evolved from work on integrating the GENIE simulations with the nd280, INGRID and 2km detector-level simulations. In recent versions of GENIE that format was renamed to 't2k_rootracker', with 'rootracker' now being a more generic, stripped-down (excudes pass-through JPARC flux info etc.) version of the T2K variance.

The 'rootracker' tree branch names, leaf types and a short description is given below. For the JNUBEAM branches please consult the corresponding documentation:

- **EvtNum** (*int*): Event number
- **EvtFlags** (*TBits**): [GENIE] Event flags.
- **EvtCode** (*TObjString**): [GENIE] A string event code.
- **EvtXSec** (*double*): [GENIE] Event cross section (in $10^{38}cm^2$).
- **EvtDXSec** (*double*): [GENIE] Differential cross section for the selected kinematics in the K^n space (in $10^{38}cm^2/$ $[K^n]$). Typically, K^n is: $\{Q^2\}$ for QEL, $\{Q^2, W\}$ for RES, $\{x, y\}$ for DIS and COH, $\{y\}$ for ve^- etc.
- **EvtWght** (*double*): [GENIE] Event weight.
- **EvtProb** (*double*): [GENIE] Event probability (given cross section, density-weighted path-length, etc).
- **EvtVtx** (*double[4]*): [GENIE] Event vertex position (x, y, z, t) in the detector coordinate system (in SI).
- **StdHepN** (*int*): [GENIE] Number of entries in the particle array.
- **StdHepPdg** (*int*): [GENIE] k^{th} particle PDG code int StdHepPdg[kNPmax];
- **StdHepStatus** (*int*): [GENIE] k^{th} particle status code int StdHepStatus[kNPmax];
- **StdHepX4** (*double [kNPmax][4]*): [GENIE] k^{th} particle 4-position (x, y, z, t) in the hit nucleus rest frame (in fm)
- **StdHepP4** (*double [kNPmax][4]*): [GENIE] k^{th} particle 4-momentum (px, py, pz, E) in the LAB frame (in GeV).

- **StdHepPolz** (*double [kNPmax][3]*): [GENIE] k^{th} particle polarization vector.
- **StdHepFd** (*int [kNPmax]*): [GENIE] k^{th} particle first-daughter index.
- **StdHepLd** (*int [kNPmax]*): [GENIE] k^{th} particle last-daughter index.
- **StdHepFm** (*int [kNPmax]*): [GENIE] k^{th} particle first-mother index.
- **StdHepLm** (*int [kNPmax]*): [GENIE] k^{th} particle last-mother index.

The following branches exist only in the 't2k_rootracker' variance:

- **NuParentPdg** (*int*): [JNUBEAM] Parent PDG code.
- **NuParentDecMode** (*int*): [JNUBEAM] Parent decay mode.
- **NuParentDecP4** (*double [4]*): [JNUBEAM] Parent 4-momentum at decay.
- **NuParentDecX4** (*double [4]*): [JNUBEAM] Parent 4-position at decay.
- **NuParentProP4** (*double [4]*): [JNUBEAM] Parent 4-momentum at production.
- **NuParentProX4** (*double [4]*): [JNUBEAM] Parent 4-position at production.
- **NuParentProNVtx** (*int*): [JNUBEAM] Parent vertex id.
- **G2NeutEvtCode** (*int*): corresponding NEUT reaction code for the GENIE event.

The following branches exist only in the 'numi_rootracker' variance[2]:

- **NumiFluxRun** (*int*): [GNUMI] Run number.
- **NumiFluxEvtno** (*int*): [GNUMI] Event number (proton on target).
- **NumiFluxNdxdz** (*double*): [GNUMI] Neutrino direction slope (dx/dz) for a random decay.
- **NumiFluxNdydz** (*double*): [GNUMI] Neutrino direction slope (dy/dz) for a random decay.
- **NumiFluxNpz** (*double*): [GNUMI] Neutrino momentum (GeV/c) along z direction (beam axis).
- **NumiFluxNenergy** (*double*): [GNUMI] Neutrino energy (GeV/c) for a random decay.

[2]More details on the GNuMI beam simulation outputs can be found at *http://www.hep.utexas.edu/~zarko/wwwgnumi/v19/*

- **NumiFluxNdxdznea** (*double*): [GNUMI] Neutrino direction slope (dx/dz) for a decay forced at center of near detector.
- **NumiFluxNdydznea** (*double*): [GNUMI] Neutrino direction slope (dy/dz) for a decay forced at center of near detector.
- **NumiFluxNenergyn** (*double*): [GNUMI] Neutrino energy for a decay forced at center of near detector.
- **NumiFluxNwtnear** (*double*): [GNUMI] Neutrino weight for a decay forced at center of near detector.
- **NumiFluxNdxdzfar** (*double*): [GNUMI] Neutrino direction slope (dx/dz) for a decay forced at center of far detector.
- **NumiFluxNdydzfar** (*double*): [GNUMI] Neutrino direction slope (dy/dz) for a decay forced at center of far detector.
- **NumiFluxNenergyf** (*double*): [GNUMI] Neutrino energy for a decay forced at center of far detector.
- **NumiFluxNwtfar** (*double*): [GNUMI] Neutrino weight for a decay forced at center of far detector.
- **NumiFluxNorig** (*int*): [GNUMI] Obsolete
- **NumiFluxNdecay** (*int*): [GNUMI] Decay mode that produced neutrino[3]
- **NumiFluxNtype** (*int*): [GNUMI] Neutrino flavor.
- **NumiFluxVx** (*double*): [GNUMI] Position of hadron/muon decay, X coordinate.
- **NumiFluxVy** (*double*): [GNUMI] Position of hadron/muon decay, Y coordinate.

[3]

- 1: K0L -> nue pi- e+
- 2: K0L -> nuebar pi+ e-
- 3: K0L -> numu pi- mu+
- 4: K0L -> numubar pi+ mu-
- 5: K+ -> numu mu+
- 6: K+ -> nue pi0 e+
- 7: K+ -> numu pi0 mu+
- 8: K- -> numubar mu-
- 9: K- -> nuebar pi0 e-
- 10: K- -> numubar pi0 mu-
- 11: mu+ -> numubar nue e+
- 12: mu- -> numu nuebar e-
- 13: pi+ -> numu mu+
- 14: pi- -> numubar mu-

- **NumiFluxVz** (*double*): [GNUMI] Position of hadron/muon decay, Z coordinate.
- **NumiFluxPdpx** (*double*): [GNUMI] Parent momentum at decay point, X - component.
- **NumiFluxPdpy** (*double*): [GNUMI] Parent momentum at decay point, Y - component.
- **NumiFluxPdpz** (*double*): [GNUMI] Parent momentum at decay point, Z - component.
- **NumiFluxPpdxdz** (*double*): [GNUMI] Parent dx/dz direction at production.
- **NumiFluxPpdydz** (*double*): [GNUMI] Parent dy/dz direction at production.
- **NumiFluxPppz** (*double*): [GNUMI] Parent Z momentum at production.
- **NumiFluxPpenergy** (*double*): [GNUMI] Parent energy at production.
- **NumiFluxPpmedium** (*int*): [GNUMI] Tracking medium number where parent was produced.
- **NumiFluxPtype** (*int*): [GNUMI] Parent particle ID (PDG)
- **NumiFluxPpvx** (*double*): [GNUMI] Parent production vertex, X coordinate (cm).
- **NumiFluxPpvy** (*double*): [GNUMI] Parent production vertex, Y coordinate (cm).
- **NumiFluxPpvz** (*double*): [GNUMI] Parent production vertex, Z coordinate (cm).
- **NumiFluxMuparpx** (*double*): [GNUMI] Repeat of information above, but for muon neutrino parents.
- **NumiFluxMuparpy** (*double*): [GNUMI] -//-.
- **NumiFluxMuparpz** (*double*): [GNUMI] -//-.
- **NumiFluxMupare** (*double*): [GNUMI] -//-.
- **NumiFluxNecm** (*double*): [GNUMI] Neutrino energy in COM frame.
- **NumiFluxNimpwt** (*double*): [GNUMI] Weight of neutrino parent.
- **NumiFluxXpoint** (*double*): [GNUMI] Unused.
- **NumiFluxYpoint** (*double*): [GNUMI] Unused.
- **NumiFluxZpoint** (*double*): [GNUMI] Unused.
- **NumiFluxTvx** (*double*): [GNUMI] Exit point of parent particle at the target, X coordinate.

- **NumiFluxTvy** (*double*): [GNUMI] Exit point of parent particle at the target, Y coordinate.
- **NumiFluxTvz** (*double*): [GNUMI] Exit point of parent particle at the target, Z coordinate.
- **NumiFluxTpx** (*double*): [GNUMI] Parent momentum exiting the target, X - component.
- **NumiFluxTpy** (*double*): [GNUMI] Parent momentum exiting the target, Y- component.
- **NumiFluxTpz** (*double*): [GNUMI] Parent momentum exiting the target, Z - component.
- **NumiFluxTptype** (*double*): [GNUMI] Parent particle ID exiting the target.
- **NumiFluxTgen** (*double*): [GNUMI] Parent generation in cascade[4].
- **NumiFluxTgptype** (*double*): [GNUMI] Type of particle that created a particle flying of the target.
- **NumiFluxTgppx** (*double*): [GNUMI] Momentum of a particle, that created a particle that flies off the target (at the interaction point), X - component.
- **NumiFluxTgppy** (*double*): [GNUMI] Momentum of a particle, that created a particle that flies off the target (at the interaction point), Y - component.
- **NumiFluxTgppz** (*double*): [GNUMI] Momentum of a particle, that created a particle that flies off the target (at the interaction point), Z - component.
- **NumiFluxTprivx** (*double*): [GNUMI] Primary particle interaction vertex, X coordinate.
- **NumiFluxTprivy** (*double*): [GNUMI] Primary particle interaction vertex, Ycoordinate.
- **NumiFluxTprivz** (*double*): [GNUMI] Primary particle interaction vertex, Z coordinate.
- **NumiFluxBeamx** (*double*): [GNUMI] Primary proton origin, X coordinate.
- **NumiFluxBeamy** (*double*): [GNUMI] Primary proton origin, Y coordinate.

[4]
- 1: Primary proton,
- 2: Particles produced by proton interaction,
- 3: Particles produced by interactions of the 2's,
- ...

- **NumiFluxBeamz** (*double*): [GNUMI] Primary proton origin, Z coordinate.
- **NumiFluxBeampx** (*double*): [GNUMI] Primary proton momentum, X - component.
- **NumiFluxBeampy** (*double*): [GNUMI] Primary proton momentum, Y - component.
- **NumiFluxBeampz** (*double*): [GNUMI] Primary proton momentum, Z - component.

6.6.2.4 The 'tracker' formats

The 'tracker'-type format is a legacy event format used by some fortran-based event generators (eg. NUANCE) and detector-level simulations (eg. SuperK's Geant3-based SKDETSIM). GENIE includes a number of 'tracker' format variations:

*** 't2k_tracker':**

This is tracker-type format with all the tweaks required for passing GENIE events into the Geant3-based SuperK detector MC. In the 't2k_tracker' files:

- The begging of event file is marked with a `$begin` line, while the end of the file is marked by an `$end` line.
- Each new event is marked with a `$genie` line. What follows is a reaction code. Since GENIE doesn't use integer reaction codes, it is writting-out the corresponding NEUT reaction code for the generated GENIE event. This simplifies comparisons between the GENIE and NEUT samples in SuperK physics analyses.
- The `$vertex` line is being used to pass the interaction vertex position in the detector coordinate system in SI units
- The `$track` lines are being used to pass minimal information on (some) initial / intermediate state particles (as expected by SKDETSIM) and all final state particles to be tracked by the detector simulation. Each `$track` line includes the particle PDG code, its energy, its direction cosines and a 'status code' (Not to be confused with GENIE's status code. The 'tracker' file status code expected by SKDETSIM is '-1' for initial state particles, '0' for stable final states and '-2' for intermediate particles).

Some further clarifications are in order here:

- K^0, $\bar{K^0}$ generated by GENIE are converted to K^0_L, K^0_S (as expected by SKDETSIM)
- Since no mother / daughter associations are stored in `$track` lines only one level of intermediates can exist (the 'primary' hadronic system). Any intermediate particles corresponding to states evolved from the 'primary' hadronic state but before reaching the 'final state' are neglected.

- The `$track` line ordering is the one expected by SKDETSIM with all the primaries, intermediates and final states grouped together.

The ‘t2k_tracker’ format includes a set of `$info` lines. They include the exact same information as the one stored ‘t2k_rootracker’ format files (complete event information generated by GENIE and JPARC / JNUBEAM flux pass-through information). This is partially redundant information (some of it was included in the minimal `$track` lines) that is not intended for pushing particles through the detector simulation. The `$info` lines are read-in by SKDETSIM and are passed-through to the DST stage so that the identical, full MC information is available for events simulated on both SuperK and the near detector complex (thus enabling global systematic studies).

A complete event in ‘t2k_tracker’ format looks-like:

```
$begin

$genie  {neut_like_event_type}
$vertex {vtxx} {vtxy} {vtxz} {vtxt}

$track  {pdg code} {E} {dcosx} {dcosy} {dcosz} {status}
$track  {pdg code} {E} {dcosx} {dcosy} {dcosz} {status}
...
$track  {pdg code} {E} {dcosx} {dcosy} {dcosz} {status}

$info {event_num} {error_code} {genie_event_type}
$info {event_xsec} {event_kinematics_xsec} {event_weight} {event_probability}
$info {vtxx} {vtxy} {vtxz} {vtxt}

$info {nparticles}
$info {idx}{pdg}{status}{fd}{ld}{fm}{lm}{px}{py}{pz}{E}{x}{y}{z}{t}{polx}{poly}{polz}
$info {idx}{pdg}{status}{fd}{ld}{fm}{lm}{px}{py}{pz}{E}{x}{y}{z}{t}{polx}{poly}{polz}
...
$info {idx}{pdg}{status}{fd}{ld}{fm}{lm}{px}{py}{pz}{E}{x}{y}{z}{t}{polx}{poly}{polz}

$info (jnubeam_parent_pdg) (jnubeam_parent_decay_mode)
$info (jnubeam_dec_px) (jnubeam_dec_py) (jnubeam_dec_pz) (jnubeam_dec_E)
$info (jnubeam_dec_x)  (jnubeam_dec_y)  (jnubeam_dec_z)  (jnubeam_dec_t)
$info (jnubeam_pro_px) (jnubeam_pro_py) (jnubeam_pro_pz) (jnubeam_pro_E)
$info (jnubeam_pro_x)  (jnubeam_pro_y)  (jnubeam_pro_z)  (jnubeam_pro_t)
$info (jnubeam_nvtx)

$end
```

6.7 Converting the XML cross section splines to ROOT

As seen before, GENIE's *gmkspl* utility writes out the computed cross section values in XML format. While this is particularly usefull format for moving data between GENIE applications / drivers, is not usefull from the perspective of a user who want to use the cross section data in different contexts

Users can use GENIE's *gspl2root* utility to convert XML cross section splines into a ROOT formats. The cross section splines (for all modeled processes) for a given initial state, can be converted into ROOT *TGraph* objects and written out in ROOT TDirectories named after the given initial state. Multiple *TDirectories* can be saved in a single file. ROOT *TGraph* can be used for numerical interpolation (using '*TGraph::Eval(double)*') so, essentially, **one can write-out all GENIE cross section 'functions' he needs into a single ROOT file**. This is obviously particularly usefull for using GENIE cross section calculations into user / analysis code.

In summary, the *gspl2root* utility syntax is:

```
shell$ gspl2root -f xml_file -p nu -t tgt [-e emax] [-o root_file] [-w]
```

where [] denotes an optional argument.

Details on the *gspl2root* options can be found below:

- **-f** Specifies the input XML cross section spline file.
- **-p** Specifies the neutrino PDG code.
- **-t** Specifies the target PDG code (format: 10LZZZAAAI).
- **-e** Specifies the maximum energy for the generated graphs.
- **-o** Specifies the output ROOT file name.
- **-w** Instructs *gspl2root* to write-out plots in a postscipt file.

The spline data written out have the energy in $GeV$and the cross section in $10^{-38}cm^2$.

Example:

1. In order to extract all ν_μ+n, ν_μ+p and ν_μ+O^{16} cross section splines from the input XML file '*mysplines.xml*', convert splines into a ROOT format and save them into a single ROOT file '*xsec.root*', type:

   ```
   shell$ gspl2root -f mysplines.xml -p 14 -t 1000000010 -o xsec.root
   shell$ gspl2root -f mysplines.xml -p 14 -t 1000010010 -o xsec.root
   shell$ gspl2root -f mysplines.xml -p 14 -t 1000080160 -o xsec.root
   ```

   A large number of graphs (one per simulated process and appropriate totals) will be generated in each case. Each set of plots is saved into its own ROOT *TDirectory* named after the specified initial state.

The stored graphs can be used for cross section interpolation. For instance, the '*xsec.root*' file generated in this example will contain a 'nu_mu_O16' *TDirectory* (generated by the last command) which will include cross section graphs for all ν_{μ}+O^{16} processes. To extract the ν_{μ}+O^{16} DIS CC cross section graph for hit u valence quarks in a bound proton and evaluate the cross section at energy E, type:

```
root[0] TFile file("xsec.root","read");
root[1] TDirectory * dir = (TDirectory*) file->Get("nu_mu_O16");
root[2] TGraph * graph = (TGraph*) dir->Get("dis_cc_p_uval");
root[3] cout << graph->Eval(E) << endl;
```

6.8 Units

GENIE is using the natural system of units ($\hbar = c = 1$) so (almost) everything is expressed in $[GeV]^n$. Notable exceptions are the event vertex (in SI units, in the detector coordinate system) and particle positions (in fm, in the hit nucleus coordinate system). Additionally, although internally all cross sections are expressed in the natural system units, values copied to certain files (eg 'rootracker'- or 'tracker'-format files) are converted to $10^{38} cm^2$ (See the corresponding documentation for these file formats).

GENIE provides an easy way for converting back and forth between its internal, natural system of units and other units. The conversion factors are included in '*$GENIE/src/Conventions/Units.h*'.

For exampe, in order to convert a cross section value returned by '*a_function()*' from the natural system of units to $10^{38} cm^2$, type:

```
double xsec = a_function() / (1E-38 * units::cm2);
```

Chapter 7

Non-neutrino GENIE modes

7.1 Introduction

[to be written]

7.2 Electron scattering mode

[to be written]

7.3 Hadron scattering mode

[to be written]

Chapter 8

Event Reweighing

8.1 Introduction

This chapter describes strategies for handling neutrino interaction uncertainties in the context of a single, self-consistent physics description. It is organized as follows: In Sec. 8.2 we describe the reweighting strategy for handling intranuclear rescattering uncertainties. In Sec. 8.4 we describe the strategy for handling neutrino cross section uncertainties. In Sec. 8.5 we present an application of the reweighting machinery.

The reweighting schemes described here are tied to the default physics choices made in GENIE (v2.4.0) and they have been implemented in the GENIE ReWeight package. As GENIE evolves, by including better-motivated theoretical models and integrating new data in its effective models[1], the reweighting schemes implemented in the ReWeight package will be transparently updated and this document will be updated accordingly.

8.2 Intranuclear hadron transport reweighing

Hadrons produced in the nuclear environment may rescatter on their way out of the nucleus, and these reinteractions significantly modify the observable distributions. The effect of hadronic reiniteractions is illustrated in Tab. 8.1 and Fig. 8.1. The sensitivity of a particular experiment to intranuclear rescattering depends strongly on the detector technology, the energy range of the neutrinos, and the physics measurement being made.

The GENIE hadron transport simulation strategy has been described in detail at a previous chapter. In the following subsections we discuss an event reweighting strategy allowing a full systematic study in the context of a physics analysis. The reweighting validation procedure will be discussed and results will be presented.

8.2.1 Reweighting strategy

Neutrino generators typically use intranuclear cascade simulations to handle the propagation of hadronic multi-particle states. At each simulation step a large

[1]The GENIE development roadmap is outlined at: http://releases.genie-mc.org

Final-	Primary Hadronic System									
State	$0\pi X$	$1\pi^0 X$	$1\pi^+ X$	$1\pi^- X$	$2\pi^0 X$	$2\pi^+ X$	$2\pi^- X$	$\pi^0\pi^+ X$	$\pi^0\pi^- X$	$\pi^+\pi^- X$
$0\pi X$	**293446**	12710	22033	3038	113	51	5	350	57	193
$1\pi^0 X$	1744	**44643**	3836	491	1002	25	1	1622	307	59
$1\pi^+ X$	2590	1065	**82459**	23	14	660	0	1746	5	997
$1\pi^- X$	298	1127	1	**12090**	16	0	46	34	318	1001
$2\pi^0 X$	0	0	0	0	**2761**	2	0	260	40	7
$2\pi^+ X$	57	5	411	0	1	**1999**	0	136	0	12
$2\pi^- X$	0	0	0	1	0	0	**134**	0	31	0
$\pi^0\pi^+ X$	412	869	1128	232	109	106	0	**9837**	15	183
$\pi^0\pi^- X$	0	0	1	0	73	0	8	5	**1808**	154
$\pi^+\pi^- X$	799	7	10	65	0	0	0	139	20	**5643**

Table 8.1: Occupancy of primary and final state hadronic systems for interactions off O^{16} computed with GENIE v2.4.0. The off-diagonal elements illustrate and quantify the topology changing effect of intranuclear rescattering.

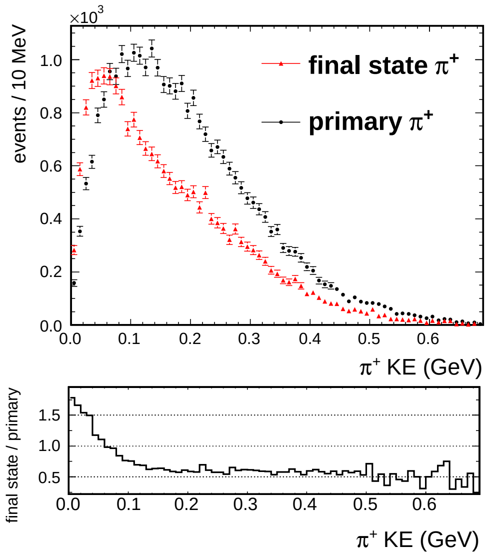

Figure 8.1: Kinetic energy spectrum of final state and primary (before rescattering) π^+ produced in $\nu_\mu Fe^{56}$ interactions at 1 GeV.

number of outcomes is accessible with the probabilities of those outcomes being conditional upon the hadron transport history up to that point. The complexity of intranuclear hadron transport makes it difficult to evaluate the probability for a generated multi-particle final state, given a primary hadronic multi-particle system, without resorting to a Monte Carlo method. Subsequently, is not possible to evaluate how that probability ought to be modified in response to changes in the fundamental physics inputs. As a result it is generally not possible to build comprehensive reweighing schemes for intranuclear hadron-transport simulations.

In this regard GENIE's INTRANUKE/hA model is unique by virtue of the simplicity of the simulation while, at the same time, it exhibiting very reliable aspects by being anchored to key hadron-nucleon and hadron-nucleus data. Its simplicity allows a rather straightforward probability estimate for the generated final state making it amenable to reweighing. A full systematic analysis of the model is therefore possible making it a unique tool in the analysis of neutrino data. The event reweighing strategy to be presented here is *specific* to GENIE's INTRANUKE/hA model. The current reweighing implementation has been tied to the physics choices made in the GENIE v2.4.0[2].

Any intranuclear hadron-transport reweighing strategy should, by *virtue of construction*, have no effect on the inclusive leptonic distributions of the reweighted sample, as illustrated in Fig. 8.2. In this paper we will be referring to that probability conservation condition as the *'unitarity constraint'*. We em-

[2]The validity of the current reweighing implementation in future versions of GENIE is dependent upon the INTRANUKE/hA changes that may be installed. The T2KReWeight package will always be updated and kept in sync with GENIE. In case of important updates a follow-up internal note will be posted.

phasize the fact that the constraint needs to hold only for unselected samples. It does not need to hold for selected samples, where the normalization is expected to vary due to the effect of the cut acceptance.

The unitarity constraint is obviously very difficult to satisfy by virtue of construction and has had a significant role in determining the reweighing strategy. Additionally, the constraint played an important role in validating the reweighing scheme and in matching exactly all physics assumptions of the original simulation. The most profound effect of weighting artifacts is to cause the unitarity constraint to be violated. We will revisit the issue of the unitarity constraint in later sections and, particularly, on the discussion of the reweighing validation.

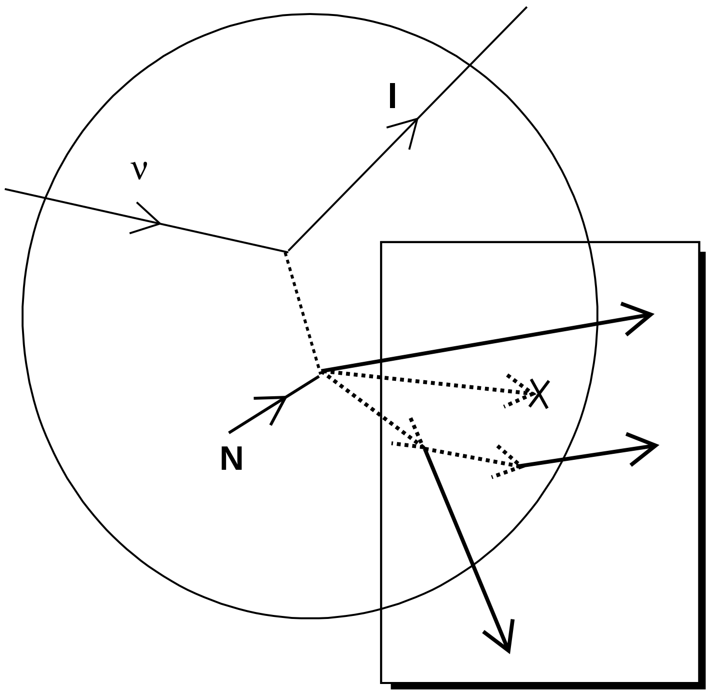

Figure 8.2: Consider the effect of modifying the intranuclear hadron-transport physics (affecting the particles within the *box*) from the perspective of an observer who is blind to the hadronic system emerging from the nucleus and measures only the primary lepton. One can easily assert that, from the perspective of that observer, the hadron-transport reweighing scheme should have no effect on the leptonic system characteristics of samples that have not been selected for hadronic system characteristics. The event weights must cancel each other so as the sum of weights is conserved, therefore maintaining the sample normalization. We will be referring to that condition as the *'unitarity constraint'*. As we will see in the reweighing validation section, the scheme discussed in this note satisfies the unitarity constraint, by *virtue of construction*, to better than 1 part in 5000.

In the reweighing strategy developed here we consider 2 kinds of physics uncertainties:

- Uncertainties in the total rescattering probability for hadrons within the target nucleus.
- Uncertainties in the relative probability of rescattering modes available to each hadron once it interacts.

These physics uncertainties are considered separately for nucleons and pions. The determination of simulation parameters linked with these physics uncertainties and the prescription for calculating event weights to account for variations in these parameters is discussed next.

8.2.1.1 Reweighting the rescattering rate

During event generation, for each hadron being propagated within the nuclear environment its rescattering probability, P^h_{rescat} (or, equivalently the survival probability, P^h_{surv}) is calculated as

$$P^h_{rescat} = 1 - P^h_{surv} = 1 - \int e^{-r/\lambda^h(\vec{r},h,E_h)} dr \tag{8.1}$$

where λ^h is the mean free path and the integral is evaluated along the hadron trajectory. The mean free path is a function of the hadron type, h, the hadron energy, E_h, and its position, $\vec{r}$, within the target nucleus and is computed as

$$\lambda^h = 1/(\rho_{nucl}(r) * \sigma^{hN}(E_h)) \tag{8.2}$$

where $\rho_{nucl}(r)$ is the nuclear density profile and $\sigma^{hN}(E_h)$ the corresponding hadron-nucleon total cross section.

During the reweighing procedure, using the positions and 4-momenta of the simulated primary hadronic system particles (that is the hadrons emerging from the primary interaction vertex before any intranuclear rescattering ever took place) we calculate the exact same hadron survival probabilities as in the original simulation. In doing so we match exactly the physics choices of the hadron transport simulation code so as to avoid weighting artifacts. More importantly:

- The reweighing code accesses the same hadron-nucleon cross section and nuclear density profile functions as the simulation code. The nuclear density profiles for ^{12}C, ^{16}O and ^{56}Fe and the nucleon-nucleon and pion-nucleon cross sections used by INTRANUKE/hA in GENIE v2.4.0 are shown in Figs. 8.3 and 8.4 respectively.
- The hadrons are being transported in steps of 0.05 fm as in the original simulation.
- Each hadron is traced till it reaches a distance of $r = N * R_{nucl} = N * R_0 * A^{1/3}$, where R_0 = 1.4 fm and N = 3.0. This allows taking into account the effect the nuclear density distribution tail has on the hadron survival probability. (For example, the nuclear radius, R_{nucl} for C^{12}, O^{16} and Fe^{56} is 3.20 fm, 3.53 fm and 5.36 fm respectively. The reweighing, as the actual simulation code, integrates Eq. 8.1 for distances up to 9.62 fm, 10.58 fm and 16.07 fm respectively. Compare these values with the nuclear density profiles shown in Fig. 8.3.)

- The nuclear density distribution through which the hadron is tracked is increased by $n * \lambda_B$, where λ_B is the de Broglie wave-length of the hadron and n is a tunable parameter (in GENIE v2.4.0, INTRANUKE/hA uses $n = 1$ for nucleons and $n = 0.5$ for pions). As explained earlier, this empirical approach is an important feature of the INTRANUKE/hA mean free path tuning, accounting for the effects of wave-like processes to the hadron survival probability which are typically not well described within the context of an INC model. The reweighing code matches that feature so as to emulate the hadron survival probabilities calculated during event generation. The effect on the nuclear density profile is shown in Fig. 8.5.

The reweighing scheme allows the mean free path, λ^h, for a hadron type h to be modified in terms of its corresponding error, $\delta\lambda^h$:

$$\lambda^h \rightarrow \lambda^{h\prime} = \lambda^h(1 + x^h_{mfp} * \delta\lambda^h/\lambda^h) \tag{8.3}$$

where $\lambda^{h\prime}$ is the modified mean free path and x_{mfp} is a tweaking knob. Then, by re-evaluating the integral in Eq. 8.1, we are able to compute the hadron survival probabilities that the simulation code would have computed, had it been using the modified mean free path. The nominal, P^h_{surv}, and tweaked, $P^{h\prime}_{surv}$, survival probabilities can be used to calculate the weight that accounts for that change in the hadron mean free path. The choice of how to weight each hadron depends critically on its intranuclear transport history. Consider the case illustrated in Fig. 8.6 where a neutrino event has 2 primary hadrons, h_1 and h_2, one of which (h_1) re-interacts while the other (h_2) escapes. Had the mean free path been larger than the one used in the simulation (and therefore, had the the interaction probability been lower) then h_1's history would have been more unlikely while, on the other hand, h_2's history would have been more likely. Therefore, in order to account for an increase in mean free path, h_1 has to be weighted down while h_2 has to be weighted up (and vice versa for a mean free path decrease). The desired qualitative behavior of single-hadron weights in response to mean free path changes is summarized in Tab. 8.2. The following weighting function exhibits the desired qualitative characteristics:

$$w^h_{mfp} = \begin{cases} \frac{1-P^{h\prime}_{surv}}{1-P^h_{surv}} & \text{if } h \text{ re-interacts} \\ \\ \frac{P^{h\prime}_{surv}}{P^h_{surv}} & \text{if } h \text{ escpapes} \end{cases} \tag{8.4}$$

where P^h_{surv} is the hadron survival probability corresponding to mean free path λ^h and $P^{h\prime}_{surv}$ is the hadron survival probability corresponding to the tweaked mean free path $\lambda^{h\prime}$.

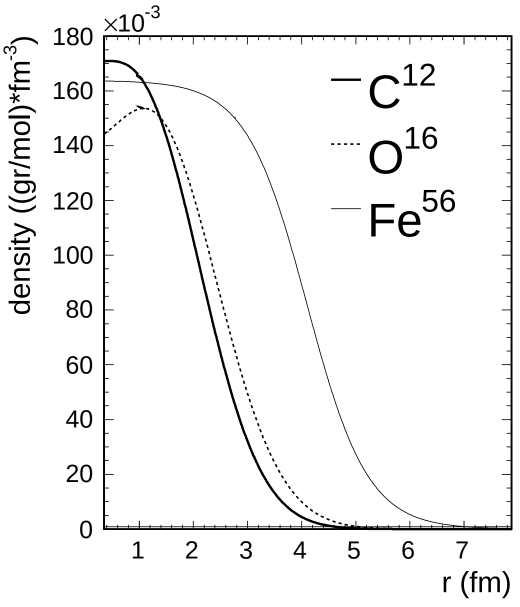

Figure 8.3: Nuclear density profiles for C^{12}, O^{16} and Fe^{56}.

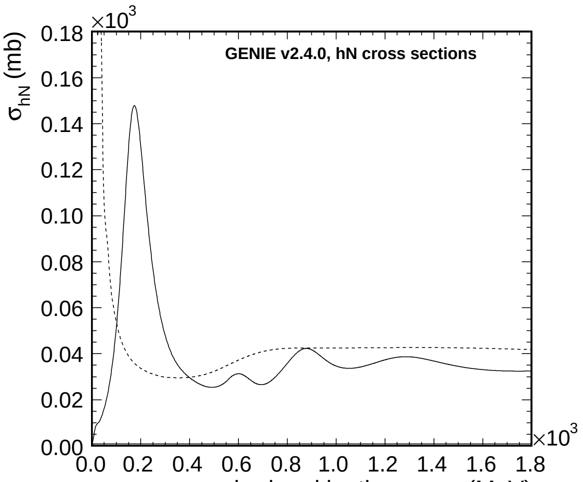

Figure 8.4: The nucleon-nucleon (dashed line) and pion-nucleon (solid line) cross sections used in INTRANUKE/hA (GENIE v2.4.0) for determining the hadron mean free path.

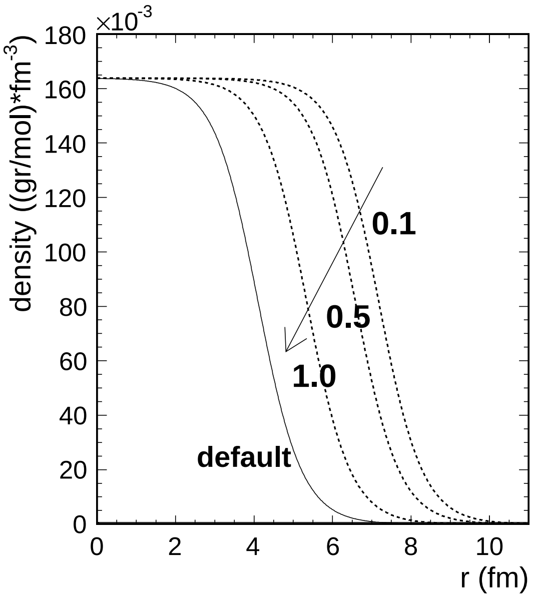

Figure 8.5: Nuclear density profiles for Fe^{56} *'stretched'* by the de-Broglie wavelength corresponding to hadrons with a momentum of 0.1 GeV, 0.5 GeV and 1.0 GeV. The default nuclear density distribution is also shown.

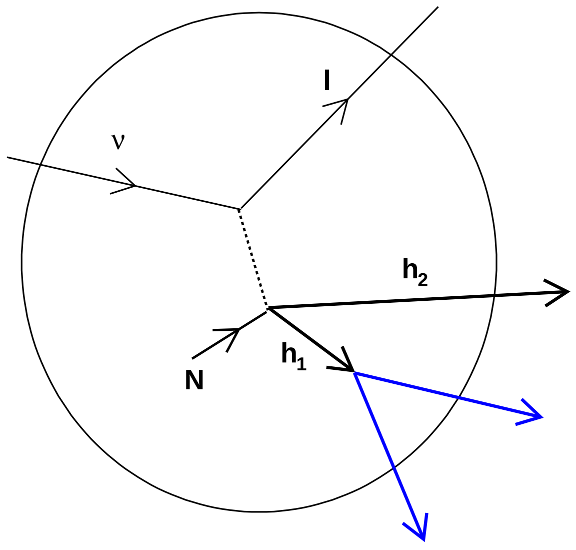

Figure 8.6: An example event with two primary hadrons, h_1 and h_2, one of which (h_1) re-interacts within the target nucleus while the other escapes (h_2). See text for a description of the weights to be assigned to each hadron if the mean free path has been tweaked.

λ^h change	P^h_{rescat}change	Weight	Weight
		(hadrons interacting)	(hadrons escaping)
$\Uparrow$	$\Downarrow$	$\Downarrow$	$\Uparrow$
$\Downarrow$	$\Uparrow$	$\Uparrow$	$\Downarrow$

Table 8.2: The intended qualitative behavior of hadron weights in response to mean free path, λ^h, changes depending on whether the simulated hadron had been rescattered or escaped. Had the mean free path been larger in reality than the one used in the simulation (and therefore, had the the interaction probability, P^h_{rescat}, been lower) then rescattered hadrons would have been over-represented in the generated sample and they would need to be weighted-down to match reality, while escaping hadrons would have been under-represented and they would need to be weighted-up. Vice versa for a mean free path decrease. See text for description of the hadron weighting functions.

8.2.1.2 Reweighting the rescattering fates

Once INTRANUKE/hA determines that a particular hadron is to be rescattered, then a host of scattering modes are available to it. We will be referring to these scattering modes as the *hadron fates*. Many fates are considered for both pions and nucleons. The fates considered here are: elastic, inelastic, charge exchange[3], absorption[4], and pion production. Each such fate may include many actual rescattering channels [5].

In order to calculate the probability of each fate INTRANUKE/hA, being an effective data-driven hadron transport MC, switches to a more macroscopic description of hadron rescattering: Rather than building everything up from hadron-nucleon cross sections, at this point in event simulation, INTRANUKE/hA determines the probability for each fate using built-in hadron-nucleus cross sections coming primarily from data. The probability for a hadron fate f is

$$P^h_f = \sigma^{hA}_f / \sigma^{hA}_{total} \tag{8.5}$$

where σ^{hA}_f is the hadron-nucleus cross section for that particular fate and σ^{hA}_{total} is the total hadron-nucleus cross section. The calculated probabilities are conditional upon a hadron being rescattered and the sum of these probabilities over all possible fates should always add up to 1. The default probability fractions for pions and nucleons in INTRANUKE/hA (GENIE v2.4.0) are shown in Fig. 8.7 and 8.8.

The generation strategy leads to a conceptually simple and technically straightforward fate reweighing strategy: The hadron-nucleus cross section for a partic-

[3]Only single charge exchange is considered

[4]Followed by emission of 2 or more nucleons with no pions in the final state. The term *'absorption'* is usually used for pions while the term *'multi-nucleon knock-out'* is used for nucleons. Here, for simplicity and in the interest of having common fate names for both pions and nucleons we will be using the term *'absorption'* for both.

[5]For example, the *'pion absorption'* fate includes rescattering modes with any of the np, pp, npp, nnp, nnpp final states

ular fate may be modified in terms of its corresponding error, $\delta\sigma_f^{hA}$ as in:

$$\sigma_f^{hA} \rightarrow \sigma_f'^{hA} = \sigma_f^{hA}(1 + x_f^h * \delta\sigma_f^{hA}/\sigma_f^{hA}) \tag{8.6}$$

where x_f is a fate tweaking knob.

It follows that the single-hadron fate weight is

$$w_{fate}^h = \sum_f \delta_{f;f'} * x_f^h * \delta\sigma_f^{hA}/\sigma_f^{hA} \tag{8.7}$$

where f runs over all possible fates {elastic, inelastic, charge exchange, absorption, pion production}, $f\prime$ is the actual fate for that hadron as it was determined during the simulation and $\delta_{f;f'}$ is a factor which is 1 if $f = f'$ and 0 otherwise.

Not all 5 hadron fates may be tweaked simultaneously. Since the sum of all fractions should add up to 1 then, at most, only 4 out of 5 fates may be tweaked directly. The 5th fate (*cushion* term) is adjusted automatically to conserve the sum. The choice of which fate to act as a cushion term is configurable. The default behavior is to have the elastic term acting as a cushion since it is large enough, over the entire range of hadron energies relevant to T2K, so as to absorb changes induced to other components. It is also the *least interesting* component in terms of its effect on the outgoing hadron and, therefore, ideal to act as a cushion term whose effect is not directly controlled.

In Fig. 8.9 we show the tweaked pion fate fraction (dashed lines) obtained by simultaneously increasing the pion production, absorption, charge exchange and inelastic cross sections by 10%. In this example the elastic component is being used as a cushion term absorbing the changes in all other terms so as to maintain the total probability. The default pion fate fractions (solid lines) are superimposed for reference.

8.2.1.3 Computing event weights

The scheme outlined above, provides a detailed prescription for calculating single-hadron weights so as to take into account the effect that modified hadron-nucleon and hadron-nucleus cross sections would have had on that hadron (w_{mfp}^h and w_{fate}^h respectively). The total single-hadron weight is

$$w^h = w_{mfp}^h * w_{fate}^h \tag{8.8}$$

The corresponding hadron transport (HT) related weight for a neutrino interaction event, w_{HT}^{evt}, is, obviously, the product of single-hadron weights

$$w_{HT}^{evt} = \prod_j w_j^h \tag{8.9}$$

where the index j runs over all the primary hadronic system particles in the event.

8.2.1.4 Computing penalty terms

A penalty term can easily be calculated from the physics tweaking knobs which can be included as nuisance parameters in physics fits. The penalty has components, penalizing deviations from the default total rescattering rate and from the default fractions of rescattering modes. It can be written as

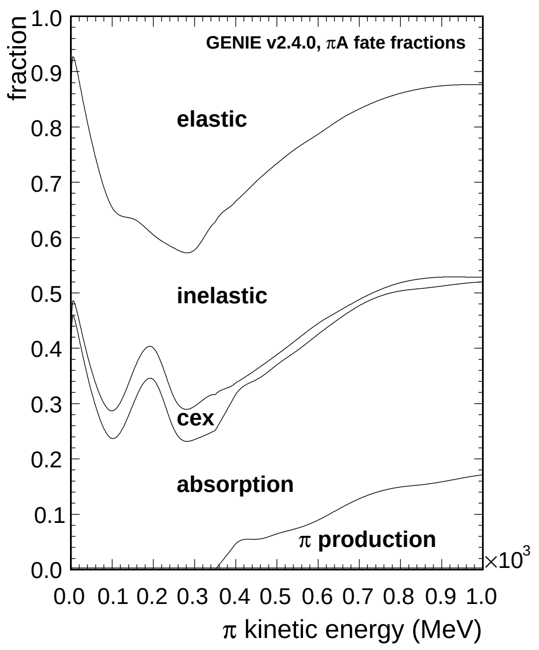

Figure 8.7: The default fate fractions for rescattered pions in INTRANUKE/hA (GENIE v2.4.0). The area that corresponds to each pion fate represents the probability for that fate as a function of the pion kinetic energy. The probabilities shown here conditional upon the pion interacting so they always add up to 1.

$$\chi^2_{penalty} = \sum_{h=\pi,N} \{(x^h_{mfp})^2 + \sum_{f \neq fc} (x^h_f)^2 + (\widehat{x^h_{fc}})^2\} \tag{8.10}$$

where the $x's$ correspond to mean free path and fate tweaking knobs for pions and nucleons The sum over fates, f, excludes the cushion term, fc, which is added separately. The reason is technical: All directly tweaked hadron-nucleus cross sections are tweaked in units of their own (typically hadron energy-dependent) uncertainty, therefore having a corresponding contribution to penalty term which is energy independent. The change in the cushion term, being forced to absorb the other changes, is not well defined in terms of its own uncertainty. Therefore, its contribution in the penalty term, $\widehat{x^{h}_{fc}}^2$, is averaged over the hadron energy range.

8.2.1.5 Unitarity expectations

This section demonstrates why both the intranuclear reweighting schemes presented earlier are expected to maintain unitarity.

In general when reweighting an event we multiply by a weight w where,

$$w = \frac{P'}{P}. \tag{8.11}$$

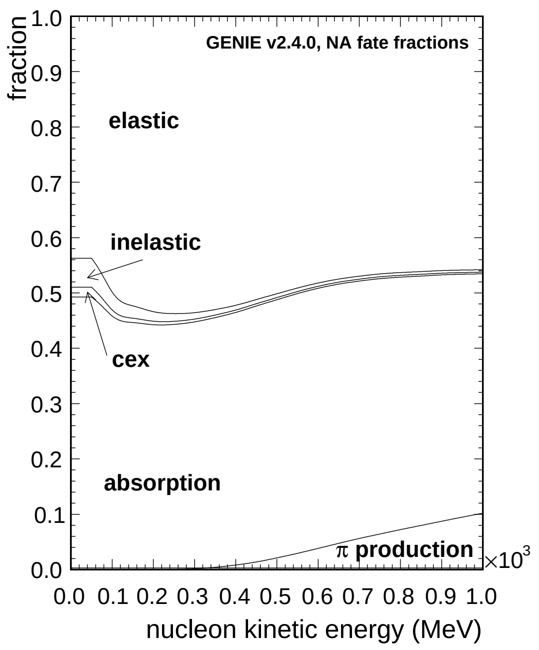

Figure 8.8: The default fate fractions for rescattered nucleons in INTRANUKE/hA (GENIE v2.4.0). The area that corresponds to each nucleon fate represents the probability for that fate as a function of the nucleon kinetic energy. The probabilities shown here conditional upon the nucleon interacting so they always add up to 1.

P and P' are the probabilities for getting that event[6], for the nominal and tweaked cases respectively, and they depend on the particular event being reweighted.

When describing processes where multiple discrete outcomes are possible then the analytical form of the above probabilities will change depending on the outcome. An example of this is the case of mean free path (rescattering rate) reweighting where the fate of an event can be divided into two categories: Those that rescattered and those that escaped the nucleus. The two forms of P in this case are,

$$P_{rescat} = 1 - e^{\frac{-x}{\lambda}} \tag{8.12}$$

and

$$P_{surv} = e^{\frac{-x}{\lambda}}. \tag{8.13}$$

Thus a hadron that rescattered will receive a weight, reflecting a change in mean free path of $\lambda \to \lambda'$, of

$$w_{rescat} = \frac{1 - e^{\frac{-x}{\lambda'}}}{1 - e^{\frac{-x}{\lambda}}} \tag{8.14}$$

[6]In this section an event is defined as the transport of a single hadron.

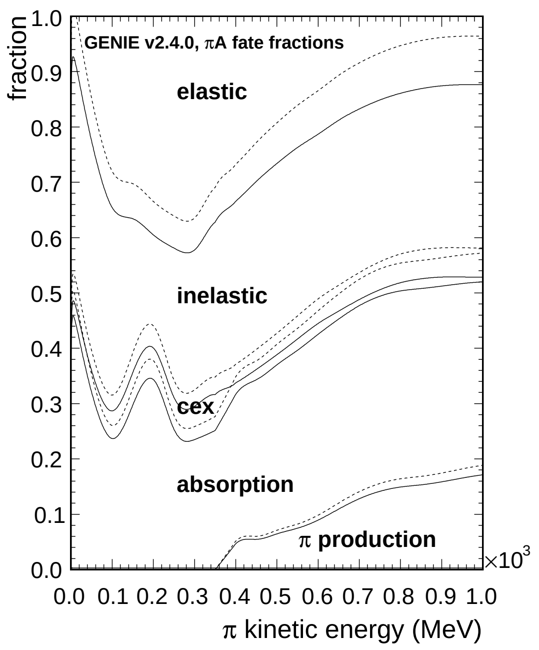

Figure 8.9: The default (solid lines) and tweaked (dashed lines) pion fate fractions. The tweaked pion fate fractions are shown for a case where the pion production, absorption, charge exchange and inelastic cross sections have been increased by 10%. Here it is the elastic cross section term that is being used as the cushion term. See text for details.

whereas one that escaped the nucleus will get a weight

$$w_{surv} = \frac{e^{\frac{-x}{\lambda'}}}{e^{\frac{-x}{\lambda}}} \tag{8.15}$$

Take the general case where there are n possible outcomes and where the i'th outcome occurs with a probability P_i. For a set of N_{tot} events one expects

$$N_i = N_{tot} \times \frac{P_i}{\sum_{j=1}^{n} P_j} \tag{8.16}$$

events corresponding to the i'th outcome.

Now consider reweighting all N_{tot} events. Events corresponding to the i'th outcome get weighted by w_i so that the after reweighting the number of events for the i'th outcome is given by

$$N_i' = w_i \times N_i. \tag{8.17}$$

Note that Eq. 8.17 holds only if we consider just the functional dependance of the weights on the weighting parameters[7]. The number of events in the new

[7]We neglect any functional dependance on kinematical quantities. This is a valid assump-

reweighted sample is given by

$$\begin{aligned} N'_{tot} &= \sum_{j=1}^{N_{tot}} w_j^{evt} \\ &= \sum_{i=1}^{n} w_i^{outcome} \times N_i \\ &= \sum_{i=1}^{n} \frac{P'_i}{P_i} \times N_i. \end{aligned}$$

Substituting Eq. 8.16 we get,

$$N'_{tot} = N_{tot} \times \frac{\sum_{i=1}^{n} P'_i}{\sum_{j=1}^{n} P_j}.$$

So if

$$\sum_{i=1}^{n} P_i = \sum_{i=1}^{n} P'_i \tag{8.18}$$

then $N'_{tot} = N_{tot}$ and unitarity is conserved.

In the case of rescattering,

$$\begin{aligned} \sum_{i=1}^{n} P_i &= P_{rescat} + P_{surv} \\ &= 1 - e^{\frac{-x}{\lambda}} + e^{\frac{-x}{\lambda}} \\ &= 1 - e^{\frac{-x}{\lambda'}} + e^{\frac{-x}{\lambda'}} \\ &= P'_{rescat} + P'_{surv} \\ &= \sum_{i=1}^{n} P'_i \end{aligned}$$

So for the rescattering scheme we expect unitarity to be a built in feature. This is also true for the fate reweighting where the cushion term ensures Eq. 8.18 is satisfied.

It is worth highlighting that when reweighting a set of generated events the P_i in Eq. 8.16 will only cancel with those in Eq. **??** if the reweighting scheme produces the same probabilities as the generator used for the original sample. This is why the unitarity constraint is sensitive to any differences between the generator and the reweighting scheme. This is also why a particular implementation of a reweighting scheme is not generator agnostic.

8.2.2 Summary of reweighting knobs

The intranuclear hadron-transport reweighing knobs are summarized in Tab. 8.3. As discussed already, there are two similar sets of parameters: one for nucleons and one for pions. The resulting parameter proliferation is well justified

tion if the density of events, defined as the number in a given volume of kinematical phase space, is high enough such that a statistically significant number of neighboring events cover a small enough volume in the kinematical phase space over which the effect of the variation in kinematical quantities is negligible.

as there may be different considerations regarding the evaluation of the uncertainty of these parameters. Additionally, systematics concerning re-interactions of nucleons and pions will have different effects on the analysis of neutrino data. Within each set

- The x_{mfp} parameter tweaks the corresponding mean free path which controls the total rescattering probability.
- The x_{cex}, x_{el}, x_{inel}, x_{abs} and x_{π} parameters tweak, respectively, the charge exchange, elastic, inelastic, absorption and pion production cross sections affecting the hadron fate fractions. Out of the 5 parameters only 4 can be set explicitly as the 5th acts as a cushion term maintaining the total hadron-nucleus interaction probability (see discussion earlier). The default cushion term is the elastic term but the choice is adjustable.

All systematic parameters are tweaked in units of their corresponding error [8].

8.2.3 Reweighting validation

The basic reweighting scheme validation procedure is outlined below. In the first part we verify that the calculated event weights behave as expected and satisfy the unitarity constraints. In the second part the power and validity of the reweighting scheme is checked explicitly by asking a simple question: Is reweighting a nominal sample using a new set of physics parameters really equivalent to a new sample generated using that same set of new parameters?

8.2.3.1 Event weight checks

As shown in the previous section both reweighting schemes are expected to maintain unitarity. Because this constraint is sensitive to any differences between the way that INTRANUKE and the ReWeight package model the transport of hadrons it was used extensively as a debugging tool during development.

Eq. 8.11 shows that the unitarity constraint is equivalent to requiring that the sum of all the weights be equal to the total number of events for a given sample. So unitarity requires that the mean of the weights is equal to 1.

When looking a distributions of weights it is useful to have an idea of the number of hadrons that are likely to escape the nucleus. Fig. 8.10 shows the distribution of distances, in mean free paths, that a hadron produced within the nucleus will have to have travelled before it has exited the nucleus. These distances are for events on ^{12}C, an intermediate size nucleus.

Fig. 8.10 shows that less than $\frac{1}{3}$ of hadrons escape the nucleus without rescattering. We expect asymmetrical weight distributions because, as is shown in Tab. 8.2, an event that rescatters will be weighted in an opposite sense to one that escapes. Thus to maintain a mean of 1 the smaller number of events that escape will have more extreme weight values to counteract the abundance of events that rescatter. Note that this is complicated by the fact that there can be multiple hadrons being transported per event with the possibility of competing effects.

[8]The evaluation of that error is an expert choice built-into the reweighing package (although easily reconfigurable).

Tweaking dial	Description	*ReWeight* package dial ID
x^N_{mfp}	Tweaks the nucleon mean free path	kSystINuke_MFPTwk_N
x^N_{cex}	Tweaks the nucleon charge exchange prob.	kSystINuke_CExTwk_N
x^N_{el}	Tweaks the nucleon elastic reaction prob.	kSystINuke_ElTwk_N
x^N_{inel}	Tweaks the nucleon inelastic reaction prob.	kSystINuke_InelTwk_N
x^N_{abs}	Tweaks the nucleon absorption prob.	kSystINuke_AbsTwk_N
x^N_{π}	Tweaks the nucleon π-production prob.	kSystINuke_PiProdTwk_N
x^{π}_{mfp}	Tweaks the π mean free path	kSystINuke_MFPTwk_pi
x^{π}_{cex}	Tweaks the π charge exchange prob.	kSystINuke_CExTwk_pi
x^{π}_{el}	Tweaks the π elastic reaction prob.	kSystINuke_ElTwk_pi
x^{π}_{inel}	Tweaks the π inelastic reaction prob.	kSystINuke_InelTwk_pi
x^{π}_{abs}	Tweaks the π absorption prob.	kSystINuke_AbsTwk_pi
x^{π}_{π}	Tweaks the π π-production prob.	kSystINuke_PiProdTwk_pi

Table 8.3: Intranuclear hadron transport reweighing knobs in the ReWeight package. Tweaking a knob, x, modifies the corresponding physics parameter, A, as $A \rightarrow A' = A(1 + x * \frac{\delta A}{A})$. Setting a knob to zero corresponds to using the nominal physics parameter. Setting the knob to +/- 1.0 corresponds to modifying the physics parameter from its nominal value by +/-1σ. In principle all 1σ errors are functions of the hadron energy. For the time being they are all set to 10% and we are working on putting in energy dependent errors, taken directly from hadron reaction data, in a near future revision.

For validation a nominal sample of 500k events of muon neutrinos at 1 GeV on C^{12} was reweighted independently for both rescattering rate and fate reweighting schemes. Tweaking dials of ± 1 corresponding to $\pm 10\%$ change of the corresponding physics parameter were used.

Figs. 8.11 and 8.12 show the weight distributions for tweaking the mean free path parameter by $+10\%$ and -10% respectively. Both distributions have a mean close to 1 to within $\sim 1/1000$ which is significantly less than the RMS of weights. This is a strong indicator that the schemes are working as intended.

Fig. 8.11 shows the expected asymmetry. In this case hadrons that rescattered have weights less than 1 as they have been made less likely due to an increase in the mean free path. Those that escaped the nucleus have weights greater than 1 reflecting the fact that they are now more likely. So we see a distribution with fewer events with a long tail for values above 1 and more events which are less spread out below 1. There is also a peak at 1 which is due to

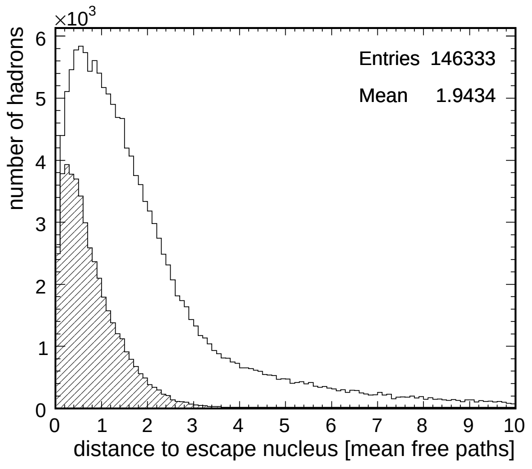

Figure 8.10: Distance a hadron produced inside the nucleus has to travel to reach the edge of the nucleus in units of mean free path. Sample is 100k events at 1 GeV on ^{12}C. The hatched region shows hadrons that escaped, 44055 out of 146333.

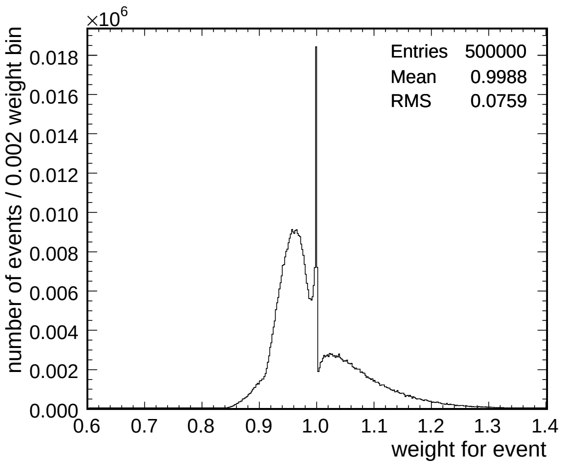

Figure 8.11: Weights distribution for tweaking both the pion and nucleon mean free paths by +10% (setting the x^{N}_{mfp} and x^{π}_{mp} tweaking dials to +1.0). Sample: 500k ν_{μ}+^{12}C events at 1 GeV.

events for which there were no hadrons to be transported out of the nucleus.

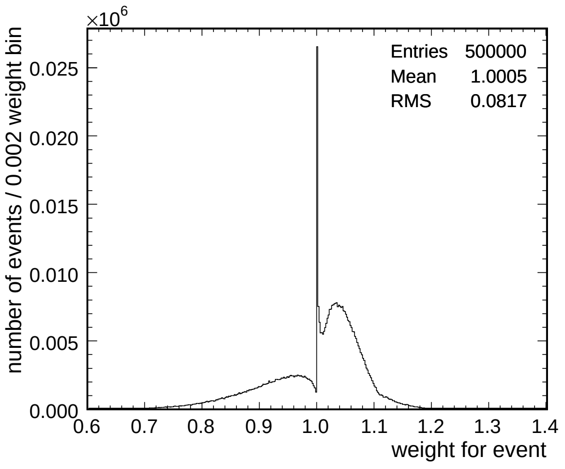

Figure 8.12: Weights distribution for tweaking both the pion and nucleon mean free paths by -10% (setting the x^{N}_{mfp} and x^{π}_{mp} tweaking dials to -1.0). Sample: 500k ν_μ+^{12}C events at 1 GeV.

Fig. 8.12 shows similar features but reflected about the origin. This is because in this case the mean free path is being decreased so that the previous effects are reversed (See Tab. 8.2).

Similar distributions but for the fates reweighting scheme are shown in Figs. 8.13 and 8.14. For these the x^{N}_{abs} and x^{π}_{abs} parameters were tweaked. For simplicity the cushion terms were chosen to be the elastic rescattering channels.

The features in Fig. 8.13 are less obvious than those for rescattering reweighting. There is a similar peak at 1 due to hadrons that escaped getting a weight of 1 alongside those events for which there were no hadrons to be transported.

There are distinct peaks at 1.1, 1.21 and a third smaller one at 1.33. These are sharp peaks because they correspond to events with absorbed hadrons. From Eq. 8.11 the weights for these cases are just the ratios of new to old cross sections which are fixed at 1.1 due to the selected value of the tweaking parameter. The 1.21 (1.33) peak corresponds to events with two (three) absorbed hadrons.

There are a series of discrete distributions mainly for values less than 1. These correspond to events that contain hadrons which interacted via elastic scattering, the cushion term in this case. The distribution of weights come from the fact that the cushion term is not fixed but instead has to vary to accommodate a fixed change in tweaking parameter alongside an energy dependent total. There are also higher order effects due to events that contain a number of hadrons with a mixture of the above two cases.

Fig. 8.14 is for the case where x^{N}_{abs} and x^{π}_{abs} were set to -1.0. As with the rescattering scheme the distribution is reflected about 1 by reversing the tweaking direction.

The weights distributions show that for both schemes the unitarity constraint respected. This is a strong indicator that the schemes are working properly. The plot in Fig. 8.15 shows the convergence on unitarity after a few thousand events.

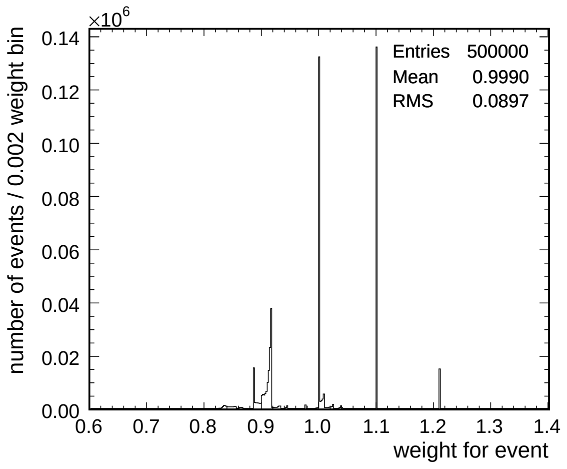

Figure 8.13: Weights distribution for the x^N_{abs} and x^{π}_{abs} tweaking dials set at $+1.0$. The mean weight is close to 1 despite the asymmetrical distribution. Sample: 500k ν_μ+^{12}C events at 1 GeV.

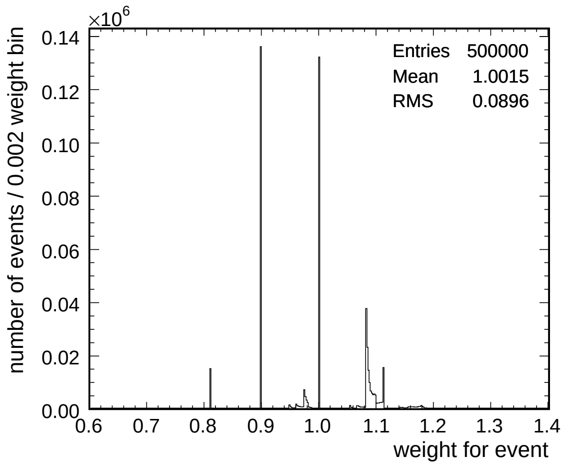

Figure 8.14: Weights distribution for x^N_{abs} and x^{π}_{abs} tweaking dials set at -1.0. Similar distribution as the $+1.0$ case but reversed. Sample: 500k ν_μ+^{12}C events at 1 GeV.

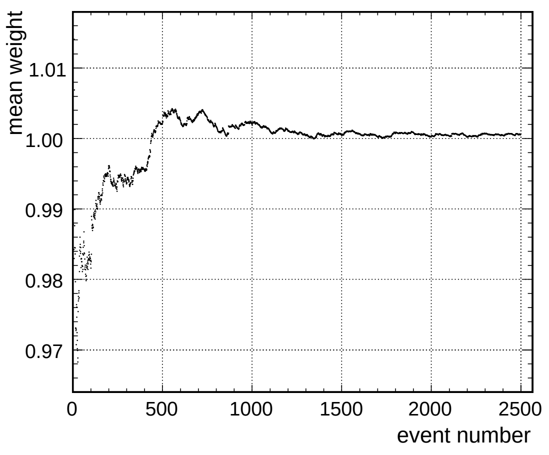

Figure 8.15: The mean weight as a function of event number showing convergence on 1 within a few thousand events. This is for a mean free path parameter tweaking knob set to +1.0. Only first 2500 events are shown.

8.2.3.2 Reweighted distribution checks

The previous section provides a strong indication that both schemes have been implemented properly. Ultimately validation comes by verifying that the reweighting scheme perfoms predictably! This is accomplished by checking whether a nominal sample reweighted using a new set of physics parameters is actually equivalent to a sample generated using the new set of parameters.

To test the overall rate reweighting two samples of 500k events of muon neutrinos on ^{12}C were generated. The first for nominal parameters used within GENIE and the second with a mean free path tweaking knob set to +1.0 (equivalent to +10% change in the mean free path). A third sample was then created from the nominal sample by reweighting with the same tweaking parameter. Fig. 8.16 shows the outgoing final state nucleon momentum distribution. It shows good agreement between the tweaked and regenerated cases. Because of the large sample size the statistical errors are small and of the same scale as the markers.

The distribution in Fig. 8.16 has the expected behavior for an increase in the mean free path as this means that hadrons are less likely to rescatter. This is the main source of final state nucleons and so we expect a deficit of final state nucleons when compared to the nominal case.

The outgoing lepton energy distribution in Fig. 8.17 shows there is no effect on the underlying leptonic distributions.

The procedure outlined above is also used to test and validate the fate reweighting scheme. In this case the tweaking parameters were x^N_{abs} and x^π_{abs} at +1.0 with the elastic component acting as the cushion term.

The distributions shown in Figs. 8.18 to 8.19 show good agreement between the reweighted and regenerated cases. They exhibit the expected behavior for an increase in the abs fate parameter. By increasing the chance of hadrons rescattering via the absorption channel the number of final state nucleons is

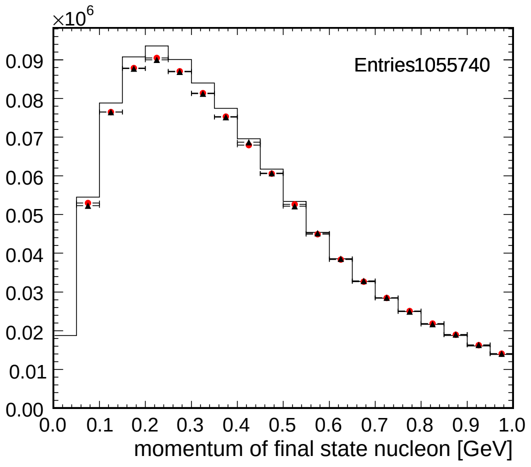

Figure 8.16: Momentum of final state nucleons for nominal (black line), regenerated (black triangle) and reweighted (red circle) samples of ν_μ+^{12}C events at 1 GeV. For regenarated and reweighted with x^N_{mfp} and x^π_{mfp} at +1.0.

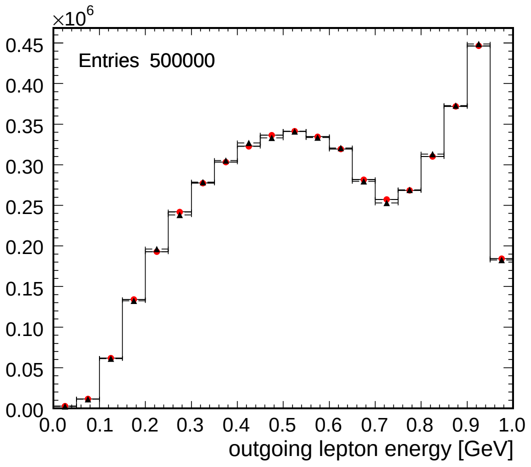

Figure 8.17: Outgoing lepton energy for nominal (black line), regenerated (black triangle) and reweighted (red circle) samples of ν_μ+^{12}C events at 1 GeV. For regenarated and reweighted with x^N_{mfp} and x^π_{mfp} at +1.0.

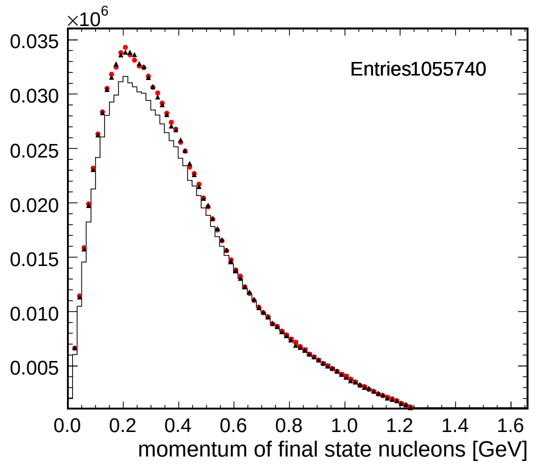

Figure 8.18: Momentum of final state nucleons for nominal (black line), regenerated (black triangle) and reweighted (red circle) samples samples of ν_μ+^{12}C events at 1 GeV. For regenerated and reweighted with x^{N}_{abs} and x^{π}_{abs} at +1.0.

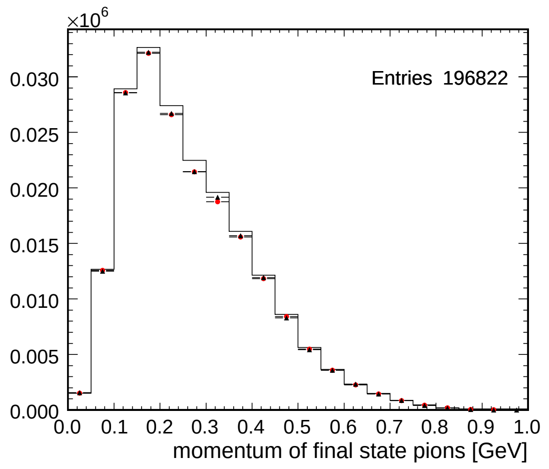

Figure 8.19: Momentum of final state π's for nominal (black line), regenerated (black triangle) and reweighted (red circle) samples of ν_μ+^{12}C events at 1 GeV. For regenerated and reweighted with x^{N}_{abs} and x^{π}_{abs} at +1.0.

increased whereas the number of final state pions is decreased due to the fact that only nucleons are produced via the absorbtion channel.

As with the rescattering scheme there is no effect on the underlying leptonic distributions. See Fig. 8.20.

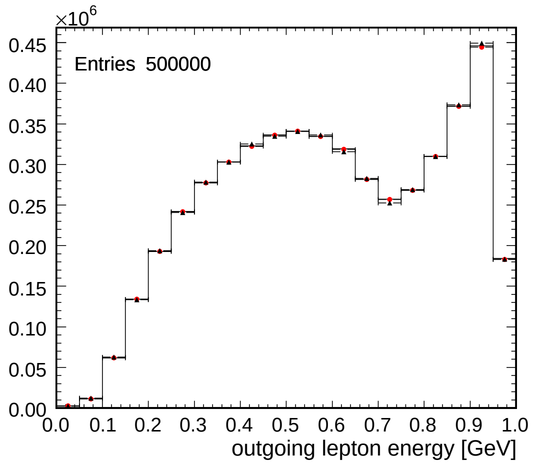

Figure 8.20: Outgoing lepton energy for nominal (black line), regenerated (black triangle) and reweighted (red circle) samples of ν_μ+^{12}C events at 1 GeV. For regenerated and reweighted with x^N_{abs} and x^π_{abs} at +1.0.

These are initial validation plots and are not meant to be a physics study into the effect of intranuclear uncertainties. As can be seen these have been for fairly simple tweaking cases and are only looking at inclusive effects summed over all outgoing final state particles.

8.3 Hadronization reweighing

No hadronization reweighting is included in the current version of GENIE.

8.4 Neutrino cross section reweighing

Unlike propagating hadronic simulation uncertainties, which is challenging as the probability for a generated multi-particle configuration is difficult to calculate analytically, propagating cross section modelling uncertainties is much simpler (since a cross section is trivially related to the event probability). However, cross section reweighting too, is not always handled correctly. One should keep in mind is that most physics uncertainties 'operate' at the form factor or structure function level which are energy independent but can be quite steep functions of the event kinematics. Therefore, the only correct way of quantifying uncertainties is to 'recover' the 'off-shell' kinematics and calculate weights by evaluating ratios of differential cross sections (See Fig. 8.21).

In the following subsections we outline the cross section model in GENIE v2.4.0 and discuss the neutrino cross section reweighting strategy in the

ReWeight package. The reweighting validation procedure and results are also presented.

8.4.1 Reweighing strategy

Unlike intranuclear hadron transport systematics, neutrino cross section systematics is straightforward to quantify using a generic reweighing scheme less strongly tied to the details of the physics modeling. Cross section reweighing is modifying the neutrino interaction probability directly and, therefore the considerations on unitarity conservation developed in the hadron transport reweighing section are not relevant here.

The neutrino event weight, w_{σ}^{evt}, to account for changes in physics parameters controlling neutrino cross sections is calculated as

$$w_{\sigma}^{evt} = (d^n\sigma'_{\nu}/dK^n)/(d^n\sigma_{\nu}/dK^n) \tag{8.19}$$

where $d^n\sigma/dK^n$ is the nominal differential cross section for the process at hand, $d^n\sigma'/dK^n$ is the differential cross section computed using the modified input physics parameters. The differential cross section is evaluated at the kinematical phase space $\{K^n\}$[9]. A critical point in implementing the cross section reweighing scheme for scattering off nuclear targets is that the correct off-shell kinematics, as used in the original simulation, must be recreated before evaluating the differential cross sections. This is trivial as long as detailed information for the bound nucleon target has been maintained by the simulation.

The neutrino cross section reweighing strategy is focused on the few-GeV energy range. Amongst the numerous GENIE cross section model parameters we are considering changes to parameters controlling the rate of QEL and RES interactions and of the 1π and 2π non-resonance background in the resonance region. Uncertainties in deep inelastic scattering and more rare processes are neglected for the time-being.

8.4.2 Summary of reweighing knobs

The neutrino cross section reweighing knobs are summarized in Tab. 8.4. At this first iteration we only included the most important parameters for the dominant processes in the few-GeV energy range. There is a singe parameter affecting the quasi-elastic cross section (the QEL axial mass, M_A^{QEL}) a single parameter affecting the resonance neutrino-production cross section (the RES axial mass, M_A^{RES}). and 16 parameters, R, controlling the 1π and 2π non-resonance background in the resonance region for various interaction modes ($\{\nu, \bar{\nu}\} \times \{n, p\} \times \{CC, NC\}$). It should be emphasized here that the apparent proliferation of tweaking parameters is brought about by the number of possible initial state, final state and weak current combinations. However, specific analyses may only consider small subsets of the R parameters. A CC$1\pi^+$ analysus,

[9]In GENIE, typically, the K^n kinematical phase space is $\{Q^2\}$ for CC quasi-elastic and NC elastic, $\{Q^2, W\}$ for resonance neutrino production, $\{x, y\}$ for deep inelastic scattering and coherent or diffractive meson production, $\{y\}$ for νe^- elastic scattering or inverse muon decay where Q^2 is the momentum transfer, W the hadronic invariant mass, x is Bjorken scaling variable and y the inelasticity. The choice is not significant. The differential cross section calculation can be mapped from the K^n to the $K^{n\prime}$ kinematic phase space through the Jacobian for the $K^n \rightarrow K^{n\prime}$ transformation.

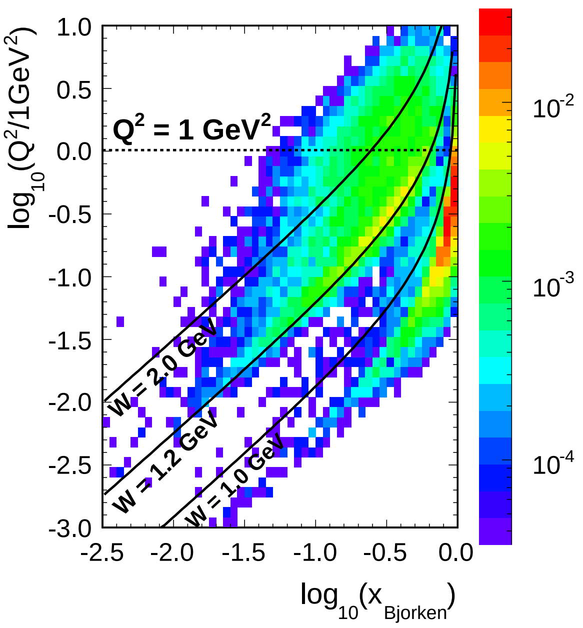

Figure 8.21: JPARC neutrino beam kinematic coverage at the nd280. Cross section uncertainties of different magnitude are appropriate for different parts of the kinematic phase space.

for example, may only consider uncertainties in the $R^{bkg}_{\nu p;CC1\pi}$ and $R^{bkg}_{\nu n;CC1\pi}$ parameters. Additionally, many of these parameters can, in fact, be correlated using isospin arguments.

8.4.3 Reweighting validation

The cross section reweighing validation procedure is the same one used for the intranuclear hadron transport reweighing: A sample is being generated with the nominal set of physics parameters (*'nominal'* sample) and is being reweighed to a tweaked set of physics parameters (*'tweak_reweighted'* sample). Then GENIE itself was reconfigured and an identical sample was generated using the tweaked set of physics parameters (*'tweak_generated'* sample). Once more we emphasize the fact that the goal of event reweighing is to emulate, approximately, what the physics simulation would have produced had the physics assumptions been somewhat different[10]. Based on the above premise, the validity of the event reweighing scheme is determined entirely on the basis of the agreement between *'tweak_reweighted'* and *'tweak_generated'* samples. As the two samples are statistically independent, large statistics have been generated to minimize statistical fluctuations.

In Fig. 8.23 we show the result of comparing the final state momentum

[10]The obvious benefit being the computational efficiency in emulating the generator predictions without rerunning the full simulation which makes event reweighing an invaluable tool for quantifying analysis systematics.

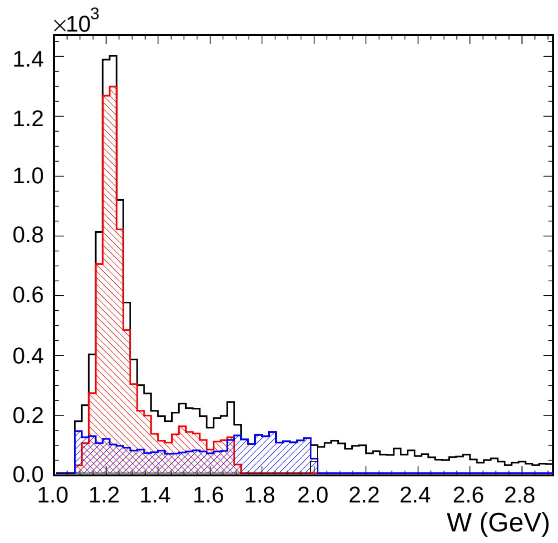

Figure 8.22: True hadronic invariant mass, W, distribution for inelastic events in nd280 (shown with the black solid line). The red hatched area shows the resonance contributions while the blue hatched area shows the contributions from the type of inelastic events dubbed in GENIE as 'transition-DIS'. The remaining contributions are coming from the 'safe-DIS' and 'low Q^2 DIS' components. Different uncertainties are associated with each component: The resonance uncertainty is of the order of 20%, while the 'transition-DIS' uncertainty is of the order of 50%. The uncertainty associated with the remaining DIS component at higher invariant masses is significantly lower (of the order of 5% at an energy of 5 GeV and lower at higher energies) and have not been included at this first iteration of deploying the reweighing tools.

distribution of 3 ν_μ QELCC samples a) a *'nominal'* sample (black line) generated with $M_A^{QEL} = 0.99$ GeV, b) a *'tweak_reweighted'* sample (red circles) reweighed (from the nominal sample) to $M_A^{QEL} = 1.1385$ GeV (+15%) c) a *'tweak_generated'* sample (black triangles) generated with $M_A^{QEL} = 1.1385$ GeV (+15% increase). The errors shown are statistical.

8.5 Example reweigting applications

The event reweighing schemes discussed in this note can be trivially integrated with neutrino-oscillation fitters with the neutrino interaction physics tweaking knobs playing the role of nuisance parameters. Another possible application is in fitting near detector distributions in order to obtain improved descriptions of the data. Correlations between the numerous parameters are automatically taken care of by virtue of the event reweighing technique.

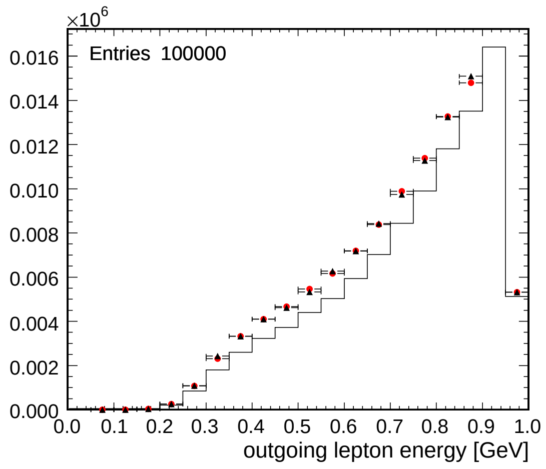

Figure 8.23: Energy of outgoing lepton for nominal (black line), regenerated (black triangle) and reweighted (red circle) samples. The reweighted and regenerated samples were such that $M_A^{QEL} = 1.1385$ GeV (+15% increase).

8.5.1 NC1π^0: Intranuclear rescattering effects

An example application of the reweighting scheme is shown in Fig. 8.24. Here the reweighting tool has been used to generate an error envelope, due to intranuclear rescattering effects, for NC1π^0[11] in the final state hadronic system) in nd280. This is an important physics uncertainty which is goingto feed into the systematic error of ν_e appearance measurements. To calculate the error envelope, a default event sample of 200k events was used. Of these only 12,538 had the required topology. To generate the error envelope the INTRANUKE/hA parameter space was scanned using the reweighting scheme. In this example the pion and nucleon INTRANUKE parameters were treated separately [12] and the uncertainties from each were combined in quadrature. In total 170 parameter configurations were scanned which without the reweighting tool would be equivalent to generating a total event sample of 34×10^6. This highlights the power of an effective reweighting scheme that allows one to reweight a smaller set of interesting events without regenerating a much larger original event sample [13].

8.6 Using the event reweighting tool

[to be written]

[11]Here the $1\pi^0$ topology refers to the observed final state, so contributions from many primary sources, with either lower or higher pion multiplicities, are to be expected (such as, for example, $\pi^0\pi^+$ primary states with an absorbed π^+ or NC elastic events where the hit nucleon re-interacts to produce a π^0)

[12]To simplify this example the mean free path parameter was assumed to be the same for both pions and nucleons and was treated as a separate systematic and added in quadrature as well.

[13]This will be particularly useful when applied to events which have been generated in a realistic detector geometry for which a user may only be interested a small subset of the total, such as a fiducial volume selection.

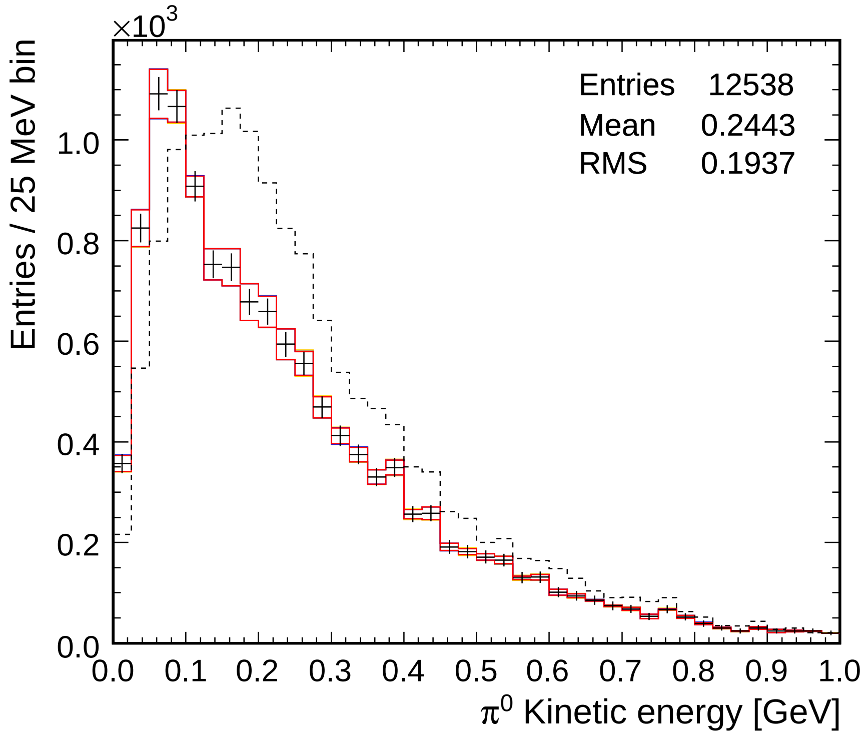

Figure 8.24: Error envelope for outgoing kinetic energy of single π^0's due to intranuclear parameter uncertainties. Original event sample of 200k events of which 12538 have a single pion topology. Dashed black line shows pion spectrum before rescattering, black cross are pion spectra after intranuclear rescattering effects and red envelope is the systematic uncertainty from intranuclear rescattering.

Physics parameter	Description	Default value	Error ($1\ \sigma$)	*ReWeight* package dial ID
M_A^{QEL}	QEL axial mass	0.990 GeV	$\sim 15\%$	kSystNuXSec_MaQEL
M_A^{RES}	RES axial mass	1.120 GeV	$\sim 20\%$	kSystNuXSec_MaRES
$R^{bkg}_{\nu p;CC1\pi}$	Controls the non-RES bkg for $\nu p\ CC1\pi$	0.1	$\sim 50\%$	kSystNuXSec_RvpCC1pi
$R^{bkg}_{\nu p;CC2\pi}$	Controls the non-RES bkg for $\nu p\ CC2\pi$	1.0	$\sim 50\%$	kSystNuXSec_RvpCC2pi
$R^{bkg}_{\nu p;NC1\pi}$	Controls the non-RES bkg for $\nu p\ NC1\pi$	0.1	$\sim 50\%$	kSystNuXSec_RvpNC1pi
$R^{bkg}_{\nu p;NC2\pi}$	Controls the non-RES bkg for $\nu p\ NC2\pi$	1.0	$\sim 50\%$	kSystNuXSec_RvpNC2pi
$R^{bkg}_{\nu n;CC1\pi}$	Controls the non-RES bkg for $\nu n\ CC1\pi$	0.3	$\sim 50\%$	kSystNuXSec_RvnCC1pi
$R^{bkg}_{\nu n;CC2\pi}$	Controls the non-RES bkg for $\nu n\ CC2\pi$	1.0	$\sim 50\%$	kSystNuXSec_RvnCC2pi
$R^{bkg}_{\nu n;NC1\pi}$	Controls the non-RES bkg for $\nu n\ NC1\pi$	0.3	$\sim 50\%$	kSystNuXSec_RvnNC1pi
$R^{bkg}_{\nu n;NC2\pi}$	Controls the non-RES bkg for $\nu n\ NC2\pi$	1.0	$\sim 50\%$	kSystNuXSec_RvnNC2pi
$R^{bkg}_{\bar{\nu} p;CC1\pi}$	Controls the non-RES bkg for $\bar{\nu} p\ CC1\pi$	0.3	$\sim 50\%$	kSystNuXSec_RvbarpCC1p
$R^{bkg}_{\bar{\nu} p;CC2\pi}$	Controls the non-RES bkg for $\bar{\nu} p\ CC2\pi$	1.0	$\sim 50\%$	kSystNuXSec_RvbarpCC2p
$R^{bkg}_{\bar{\nu} p;NC1\pi}$	Controls the non-RES bkg for $\bar{\nu} p\ NC1\pi$	0.3	$\sim 50\%$	kSystNuXSec_RvbarpNC1p
$R^{bkg}_{\bar{\nu} p;NC2\pi}$	Controls the non-RES bkg for $\bar{\nu} p\ NC2\pi$	1.0	$\sim 50\%$	kSystNuXSec_RvbarpNC2p
$R^{bkg}_{\bar{\nu} n;CC1\pi}$	Controls the non-RES bkg for $\bar{\nu} n\ CC1\pi$	0.1	$\sim 50\%$	kSystNuXSec_RvbarnCC1p
$R^{bkg}_{\bar{\nu} n;CC2\pi}$	Controls the non-RES bkg for $\bar{\nu} n\ CC2\pi$	1.0	$\sim 50\%$	kSystNuXSec_RvbarnCC2p
$R^{bkg}_{\bar{\nu} n;NC1\pi}$	Controls the non-RES bkg for $\bar{\nu} n\ NC1\pi$	0.1	$\sim 50\%$	kSystNuXSec_RvbarnNC1p
$R^{bkg}_{\bar{\nu} n;NC2\pi}$	Controls the non-RES bkg for $\bar{\nu} n\ NC2\pi$	1.0	$\sim 50\%$	kSystNuXSec_RvbarnNC2p

Table 8.4: Neutrino cross section reweighing knobs in ReWeight package. Tweaking a knob, x, modifies the corresponding physics parameter, A, as $A \rightarrow A' = A(1 + x * \frac{\delta A}{A})$. Setting a knob to zero corresponds to using the nominal physics parameter. Setting the knob to +/- 1.0 corresponds to modifying the physics parameter from its nominal value by +/-1σ.

Chapter 9

Validation tools

9.1 Introduction

[to be written]

9.2 Comparisons with the world neutrino cross section data

[to be written]

9.3 Comparisons with F2 and xF3 world data

[to be written]

9.4 Sum rule tests

[to be written]

9.5 Hadronization benchmark tests

[to be written]

9.6 Intranuclear rescattering benchmark tests

[to be written]

9.6.1 ‘Merenyi’ test

[to be written]

9.6.2 ‘hA’ tests

[to be written]

9.7 Cross-release and integrity checks

[to be written]

9.8 NuValidator

What is NuValidator?

[to be written]

Status

[to be written]

Starting the graphical user interface (GUI)

To start the GUI type:

```
shell$ gnuvld_gui &
```

Bootstraping the data-base

From the GUI:

1. Cclick 'Database' → 'Connect' to enter the data-base information using the data-base connection dialog.
2. Click 'Database' → 'Bootstrap dbase'.

Alternatively, use the *gnuvld_ dbbootstrap* GENIE utility. Type:

```
shell$ gnuvld_dbbootstrap -h host -d dbase -u user -p passwd
```

How do I fill-in the data-base?

From the GUI:

1. Click 'File' → 'Open XML' and select the input XML data file.
2. Click 'Database' → 'Connect' to enter the data-base information using the data-base connection dialog.
3. Click 'Database' →'Upload XML file to dbase'.

Alternatively, use the *gnuvld_ dbupload* GENIE utility. Type:

```
shell$ gnuvld_dbupload -f /path//xml_file -h host -d dbase -u user -p passwd
```

Using the GUI

[to be written]

Chapter 10

For Advanced Users: Extending GENIE

10.1 Introduction

10.2 Calling GENIE algorithms directly

GENIE provides a host of event generation applications and utilities and most users will only ever interact with these. It is only for the most advanced GENIE uses-cases that one may need to access and run algorithms directly. This is typically a 4-step process, as outlined below:

1. Get an algorithm factory (*AlgFactory*) instance. The algorithm factory provides access to configured instances of all GENIE algorithms.

   ```
   AlgFactory * algf = AlgFactory::Instance();
   ```

2. Request a concrete algorithm from the factory. Each algorithm is uniquely specified by its name and the name of its configuration parameter set.

   ```
   const Algorithm * alg_base = algf->GetAlgorithm(“name”, “config”);
   ```

3. Type-cast *Algorithm* to the specific algorithmic interface (*XyzI*) being implemented. For example, for cross section algorithms type-cast to *XSecAlgorithmI*, for hadronization models to *HadronizationModelI*, for strucrure function models to *DISStructureFuncModelI*, for event generation modules to *EventRecordVisitorI* etc (please consult the GENIE doxygen code reference for a full list of possibilities).

   ```
   const XzyI * alg = dynamic_cast<const XyzI *>(alg_base);
   ```

4. Prepare the algorithm inputs and run it (please consult GENIE doxygen code reference for documentation on each algorithmic interface).

Example 1

The following example shows how to get the Rein-Sehgal resonance neutrino-production model, calculate the differential cross section $d^2\sigma/dWdQ^2$ for $\nu_\mu + n$ (bound in Fe^{56}) $\rightarrow \mu^- + \mathrm{P11(1440)}$ at E_ν =2.4 GeV, W=1.35 GeV, Q^2=1.1 GeV^2 and then calculate the integrated cross section at the same energy:

```
{
  ...

  // get the algorithm factory
  AlgFactory * algf = AlgFactory::Instance();

  // get the cross section algorithm
  const Algorithm * algbase =
     algf->GetAlgorithm("genie::ReinSeghalRESPXSec", "Default"));
  const XSecAlgorithmI * xsec_model =
     dynamic_cast<const XSecAlgorithmI *> (algbase);

  // prepare the cross section algorithm inputs
  Interaction * interaction
          = Interaction::RESCC(kPdgTgtFe56,kPdgNeutron,kPdgNuMu);
  interaction->InitStatePtr()->SetProbeE(2.4);
  interaction->KinePtr()->SetW(1.35);
  interaction->KinePtr()->SetQ2(1.1);
  interaction->ExclTagPtr()->SetResonance(kP11_1440);

  // calculate d2sigma/dWdQ2 differential cross section
  // (in 1E-38 cm^2 / GeV^3)
  double diff_xsec = xsec_model->XSec(
          interaction, kPSWQ2fE) / (1E-38 * units::cm2);

  // get the integrated cross section
  // (in 1E-38 cm^2)
  double intg_xsec = xsec_model->Integral(
          interaction) / (1E-38 * units::cm2);

  ...
}
```

10.3 Adding new flux drivers

[to be written]

10.4 Adding new event generation applications

[to be written]

10.5 Adding new cross section algorithms

[to be written]

10.6 Adding new event generation modules

[to be written]

10.7 Adding new event generation threads

[to be written]

10.8 Plugging-in to the message logging system

The message logging system is based on the *log4cpp* library. GENIE provides the *Messenger* class which enforces common formatting for messages emitted by GENIE classes and provides an easier interface to the log4cpp library. Messages are sent using one of the

- LOG(stream,priority),
- LOG_FATAL(stream),
- LOG_ALERT(stream),
- LOG_CRIT(stream),
- LOG_ERROR(stream),
- LOG_WARN(stream),
- LOG_NOTICE(stream),
- LOG_INFO(stream)
- LOG_DEBUG(stream)

Messenger macros as shown in 1. Each message is assigned a priority level (see Table 10.1) that can be used for message filtering using the

```
void genie::Messenger::SetPriorityLevel(const char * stream log4cpp::Priority::Value priority)
```

method as shown in 1. Each message is 'decorated' with its time stamp, its priority level, its stream name and the name space / class name / method name / line of code from where it was emitted

```
time priority stream name : <method signature (line of code)> : actual message
```

For example:

```
10891167 ERROR Config:<bool genie::ConfigPool::LoadXMLConfig() (100)>: Parsing failed
```

Message Priority Levels
pFATAL
pALERT
pCRIT
pERROR
pWARN
pNOTICE
pINFO
pDEBUG

Table 10.1: Priority levels in GENIE / log4cpp shown in decreasing importance.

Algorithm 1 Example use of the GENIE / log4cpp message logging.

```
{
    ...
    LOG("stream-name", pFATAL) << " a fatal message";
    LOG("stream-name", pERROR) << " an error message";
    LOG("stream-name", pWARN) << " a warning";

    // alternative ways to send messages
    LOG_ERROR("stream-name") << " another error message";
    LOG_WARN("stream-name") << " another warning";
    ...
    Messenger * msg = Messenger::Instance(); // get a messenger instance
    ...
    msg->SetPriorityLevel("stream-name",pERROR); // set message threshold to 'ERROR'
    ...
    LOG("stream-name", pALERT) << " an alert – passes the message thershold";
    LOG("stream-name", pDEBUG) << " a debug message – filtered / not shown";
    ...
}
```

Chapter 11

Special topics, FAQs and Troubleshooting

Deciding which particles to decay

GENIE attempts to simulate the complex physics within the nuclear environment and, by default, it considers that every particle which escapes the target nucleus has left its realm. It is the responsibility of the detector simulation to handle particles that propagate more than a few fermis before decaying. GENIE, for example, in its default mode, will not decay charmed hadrons. If, like many others, you think that these are "short-lived" particles GENIE ought to decay then consider this: If a C^{12} nucleus was as big as the Earth, then these particles would decay more than a light year away ($c\tau_0(\Lambda_c^+)/(C^{12} radius) \sim 2 \times 10^{10}$, $c\tau_0(D_s)/(C^{12} radius) \sim 5 \times 10^{10}$, etc). Similarly, GENIE won't decay τ leptons. The default GENIE settings are appropriate as we do not want to be making any assumption regarding the user's detector technology and its ability to detect these short tracks. (Decaying τ leptons is obviously not desirable for an emulsion detector).

One can easily override the default GENIE choices by setting a series of "`DecayParticleWithCode=i`" flags at the '`$GENIE`*/config/UserPhysicsOptions.xml*' configuration file.

For example, to enable decays of τ^- leptons (PDG code = 15), one needs to change:

```
<param type=''bool'' name=''DecayParticleWithCode=15''> false </param>
```

to:

```
<param type=''bool'' name=''DecayParticleWithCode=15''> true </param>
```

Inhibiting decay channels

By default, GENIE does not inhibit any kinematically allowed channel. However, for certain studies, a user may wish to inhibit certain uninteresting decay channels in order to speed up event generation. This can be done by setting a series of "`InhibitDecay/Particle=i,Channel=j`" configuration options at the '`$GENIE`*/config/UserPhysicsOptions.xml*' file.

To figure out the decay channel code numbers use the *print_ decay_ channels.C* script in '`$GENIE`*/src/contrib/misc/*' (GENIE uses the ROOT '*TDecayChannel*' IDs).

For example, to inbibit the τ^- lepton (PDG code = 15) $\tau^- \rightarrow \nu_\tau e^- \bar{\nu}_e$ decay channel (decay channel ID = 0), one needs to type:

```
<param type=”bool” name=InhibitDecay/Particle=15,Channel=0”> true </param>
```

GENIE random number periodicity.

GENIE is using ROOT's Mersenne Twistor random number generator with periodicity of 10^{6000}. See the ROOT *TRandom3* class for details. In addition GENIE is structured to use several random number generator objects each with its own "independent" random number sequence (see discussion in ROOT *TRandom* class description). GENIE provides different random number generators for different types of GENIE modules: As an example, *RandomGen::RndHadro()* returns the generator to by used in hadronization models, *RandomGen::RndDec()* returns the generator to be used by decayers, *RandomGen::RndKine()* returns the generator to be used by kinematics generators, *RandomGen::RndFsi()* returns the generator to be used by intranuclear rescattering MCs and so on... (see *RandomGen* for the list of all generators). This is an option reserved for the future as currently all modules are passed the same random number generator (no problems with the generator periodicity have been found or reported so far).

Making user-code conditional on the GENIE version number.

This can be done in similar way as with ROOT by including '`$GENIE`*/src/Conventions/GVersion.h*' (this header files generated automatically during the GENIE installation). If, for example, one wants to do something different before/after version 2.16.22, then simply type:

```
#if __GENIE_RELEASE_CODE__ >= GRELCODE(2,16,22)
...
#else
...
#endif
```

NuValidator connection to the MySQL data-base.

On every single instance that a user had difficulties getting NuValidator to connect to the MySQL data-base it invariably turned out that the user's ROOT installation was not properly configured for MySQL. If you run into similar problems, then try connecting to your MySQL server directly from a ROOT interactive session to make sure that everything is properly configured. For example, try the following sequence:

Connect to the 'test' data-base.

```
root [0] sql = TSQLServer::Connect("mysql://localhost/test","","");
```

If you were connected succesfully get some info for the data-base.

```
root [1] cout << sql->GetHost() << endl;
localhost
root [2] cout << sql->GetPort() << endl;
0
root [3] cout << sql->GetDBMS() << endl;
MySQL
```

Create a new table called 'mytest' into the 'test' data-base.

```
root [4] sql->Query("create table mytest (I INT, J INT)");
```

Insert 3 rows.

```
root [5] sql->Query("insert into mytest values (1,1)");
root [6] sql->Query("insert into mytest values (2,3)");
root [7] sql->Query("insert into mytest values (5,6)");
```

Query the 'test' data-base for the 'mytest' table contents.

```
root [8] TSQLResult * result = sql->Query("select * from mytest");
```

Print the number of rows.

```
root [9] cout << result->GetRowCount() << endl;
3
```

Get the 1st row'(1,1)' and print the fields

```
root [10] TSQLRow * row = result->Next();
root [11] cout << row->GetField(0) << endl;
1
root [12] cout << row->GetField(1) << endl;
1
```

If there is a mis-configuration problem then it will show-up early on. Read the ROOT installation instruction and try to configure ROOT with MySQL properly.

Screen resolution too poor for the NuValidator GUI.

The NuValidator GUI windows may appear too large to be usable if the screen resolution is poor (eg 800 x 600). In this case some of the main GUI window frames can be hidden or undocked and overlayed so that the main window takes much less space. To undock frames use the options under the 'View' menu.

Appendix A

Summary of Important Physics Parameters

A large set of physics parameters controls the models described above. For convenience all these parameters are defined at a single configuration file, *'$GENIE/config/UserPhysicsOptions.xml'*. These parameters are listed below.

Grp.	Physics parameter	Default value	GENIE parameter name
Basic	V_{ud}	0.97377	`CKM-Vud`
	V_{us}	0.2257	`CKM-Vus`
	V_{cd}	0.230	`CKM-Vcd`
	V_{cs}	0.957	`CKM-Vcs`
	G_F	1.16639E-5 GeV^{-2}	
	θ_c	0.22853207	`CabbiboAngle`
	θ_w	0.49744211	`WeinbergAngle`
	m_{charm}	1.430 GeV	
	μ_p	2.7930	`AnomMagnMoment-P`
	μ_n	-1.913042	`AnomMagnMoment-N`
NC-EL	M_A	0.990 GeV	`EL-Ma`
	M_V	0.840 GeV	`EL-Mv`
	η_{axial}	0.12	`EL-Axial-Eta`
L/S QEL-CC	M_A	0.990 GeV	`QEL-Ma`
	M_V	0.840 GeV	`QEL-Mv`
	$F_A(Q^2 = 0)$	-1.2670	`QEL-FA0`
R/S RES	Ω	1.05	
	Z	0.762	
	M_A	1.120 GeV	
	M_V	0.840 GeV	
	Resonances (mass and width in MeV)	P33(1232;120), S11(1535;150), D13(1520;120), S11(1650;150), D13(1700;100), D15(1675;150), S31(1620;150), D33(1700;300), P11(1440;350), P13(1720;150), F15(1680;130), P31(1910;250), P33(1920;200), F35(1905;350), F37(1950;300), P11(1710;100)	
	Coherence	false	
B/Y DIS	A	0.538	
	B	0.305	
	C_s^u	0.363	
	C_s^d	0.621	
	C_{v1}^u	0.291	
	C_{v2}^u	0.189	
	C_{v1}^d	0.202	
	C_{v2}^d	0.255	
	X_0	-0.00817	
	X_1	0.0506	
	X_2	0.0798	
	PDF Q^2_{min}	0.800 GeV^2	
	Uncorr. PDF set	GRV98LO	
	Nuclear mod.?	true	
	Whitlow R (F_L)?	true	

Grp.	Physics parameter	Default value	GENIE parameter name
R/S COH	M_A	1.000 GeV	
	R_0	1.000 fm	
	$Re/ImAmpl$	0.300	
	Mod. PCAC?	true	
Transition Region	$R(\nu p; CC; n = 2)$	0.100	
	$R(\nu p; CC; n = 3)$	1.000	
	$R(\nu p; NC; n = 2)$	0.100	
	$R(\nu p; NC; n = 3)$	1.000	
	$R(\nu n; CC; n = 2)$	0.300	
	$R(\nu n; CC; n = 3)$	1.000	
	$R(\nu n; NC; n = 2)$	0.300	
	$R(\nu n; NC; n = 3)$	1.000	
	$R(\bar{\nu} p; CC; n = 2)$	0.300	
	$R(\bar{\nu} p; CC; n = 3)$	1.000	
	$R(\bar{\nu} p; NC; n = 2)$	0.300	
	$R(\bar{\nu} p; NC; n = 3)$	1.000	
	$R(\bar{\nu} n; CC; n = 2)$	0.100	
	$R(\bar{\nu} n; CC; n = 3)$	1.000	
	$R(\bar{\nu} n; NC; n = 2)$	0.100	
	$R(\bar{\nu} n; NC; n = 3)$	1.000	
	W_{cut}	1.7 GeV	
BBA2005	$G_{ep}(a0)$	1.	
	$G_{ep}(a1)$	-0.0578	
	$G_{ep}(a2)$	0.	
	$G_{ep}(b1)$	11.100	
	$G_{ep}(b2)$	13.60	
	$G_{ep}(b3)$	33.00	
	$G_{ep}(b4)$	0.	
	$G_{\mu p}(a0)$	1.	
	$G_{\mu p}(a1)$	0.1500	
	$G_{\mu p}(a2)$	0.	
	$G_{\mu p}(b1)$	11.100	
	$G_{\mu p}(b2)$	19.600	
	$G_{\mu p}(b3)$	7.540	
	$G_{\mu p}(b4)$	0.	
	$G_{en}(a0)$	0.	
	$G_{en}(a1)$	1.250	
	$G_{en}(a2)$	1.30	
	$G_{en}(b1)$	-9.86	
	$G_{en}(b2)$	305.0	
	$G_{en}(b3)$	-758.0	
	$G_{en}(b4)$	802.0	
	$G_{\mu n}(a0)$	1.	
	$G_{\mu n}(a1)$	1.810	
	$G_{\mu n}(a2)$	0.	
	$G_{\mu n}(b1)$	14.100	
	$G_{\mu n}(b2)$	20.70	
	$G_{\mu n}(b3)$	68.7	
	$G_{\mu n}(b4)$	0.	

Table A.1: [to be written]

Grp.	Physics parameter	Default value	GENIE parameter name
a	p_{cutoff}	0.5 GeV	
Removal Energy (GeV)	Li6	0.017	
	C12	0.025	
	O16	0.027	
	Mg24	0.032	
	Ca40	0.028	
	Fe56	0.036	
	Ni58	0.036	
	Pb208	0.044	
	other		
k_F (GeV)	Li6	(p) 0.169	
		(n) 0.169	
	C12	(p) 0.221	
		(n) 0.221	
	O16	(p) 0.225	
		(n) 0.225	
	Mg24	(p) 0.235	
		(n) 0.235	
	Si28	(p) 0.239	
		(n) 0.239	
	Ar40	(p) 0.242	
		(n) 0.259	
	Ca40	(p) 0.251	
		(n) 0.251	
	Fe56	(p) 0.251	
		(n) 0.263	
	Ni58	(p) 0.257	
		(n) 0.263	
	Sn120	(p) 0.245	
		(n) 0.274	
	Ta181	(p) 0.247	
		(n) 0.281	
	Pb208	(p) 0.245	
		(n) 0.283	

Table A.2: Default values for parameters controlling the Fermi Gas model implementation in GENIE.

Appendix B

Installation instructions for beginners

Installing 3rd party software

The following dependencies need to be installed, in the following order.

LOG4CPP

Before installing log4cpp

Check whether log4cpp is already installed at your system. The library filename contains liblog4cpp, so if you cannot find a file with a filename containing liblog4cpp, then you probably do not have the software installed.

Getting the source code

Download the source code from the sourceforge anonymous CVS repository (when prompted for a password, simply hit enter):

```
shell$ cd /dir/for/external/src/code
shell$ cvs -d :pserver:anonymous@log4cpp.cvs.sourceforge.net:/cvsroot/log4cpp login
shell$ cvs -d :pserver:anonymous@log4cpp.cvs.sourceforge.net:/cvsroot/log4cpp -z3 co
log4cpp
```

Configuring and building

Enter the log4cpp directory and run '`autogen`' and '`configure`'. Replace [location] with the installation directory of your choice; you cannot install it in the same directory as the source (where you are now). You can choose not to use the '`--prefix`' tag, in which case the default install directory is '*/usr/local*'.

```
shell$ cd log4cpp
shell$ ./autogen.sh
shell$ ./configure --prefix=[location]
```

What's left is to run '`make`' and '`make install`'. If make install gives you an error while copying or moving files stating that the files are identical, then you probably choose the source folder as your install folder in the above configure step. Rerun configure with a different location (or simply leave the '`--prefix`' option out for the default).

```
shell$ make
shell$ make install
```

Notes:

- Alternatively, you may install pre-compiled binaries. For example, if you are using 'yum' on LINUX then just type:
  `shell$ yum install log4cpp`
  On MAC OS X you can do the same using 'DarwinPorts':
  `shell$ sudo port install log4cpp`

LIBXML2

Before installing libxml2

Check whether libxml2 is already installed at your system - most likely it is. Look for a libxml2.* library (typically in '*/usr/lib*') and for a libxml2 include folder (typically in '*/usr/include*').

Getting the source code

Download the source code from the GNOME subversion repository:

```
shell$ cd /dir/for/external/src/code
shell$ svn co https://svn.gnome.org/svn/libxml2/trunk libxml2
```

Alternatively, you download the code as a gzipped tarball from:
http://xmlsoft.org/downloads.html.

Configuring and building

```
shell$ cd libxml2
shell$ ./autogen.sh --prefix=[location]
shell$ make
shell$ make install
```

Notes:

- Alternatively, you may install pre-compiled binaries. For example, if you are using 'yum' on LINUX then just type:
  `shell$ yum install libxml2`
  On MAC OS X you can do the same using 'DarwinPorts':
  `shell$ sudo port install libxml2`

LHAPDF

Getting the source code

Get the LHAPDF code (and PDF data files) from *http://projects.hepforge.org/lhapdf/*. The tarball corrsponding to version *'x.y.z'* is named *'lhapdf-x.y.z.tar.gz'*.

```
shell$ mv lhapdf-x.y.z.tar.gz /dir/for/external/src/code
shell$ cd /directory/to/download/external/code
```

```
shell$ tar xzvf lhapdf-x.y.z.tar.gz
```

Configuring and building

```
shell$ cd lhapdf-x.y.z/
shell$ ./configure --prefix=[location]
shell$ make
shell$ make install
```

PYTHIA6

Installation of PYTHIA6 is simplified by using a script provided by Robert Hatcher. A version of Robert's script is also kept at the GENIE web site at: *http://hepunx.rl.ac.uk/∼candreop/generators/GENIE/etc/installation/*

You can run the script (please, also read its documentation) as:

```
shell$ source build_pythia6.sh [version]
```

For example, in order to download and install version 6.4.12, type:

```
shell$ source build_pythia6.sh 6412
```

ROOT

Getting the source code

Get the source code from the ROOT subversion repository. To get the development version, type:

```
shell$ cvs co http://root.cern.ch/svn/root/trunk root
```

To get a specific version '*x.y.z*', type:

```
shell$ cvs co http://root.cern.ch/svn/root/tags/vx-y-z root
shell$ cvs co http://root.cern.ch/svn/root/tags/v5-22-00 root
```

See *http://root.cern.ch/drupal/content/downloading-root/*

Configuring and building

```
shell$ export ROOTSYS=/path/to/install_root
shell$ cd $ROOTSYS
shell$ ./configure [arch] [other options] --enable-pythia6 --with-pythia6-libdir=$PYTHIA6

shell$ make
```

Testing

Accessing root is an easy test to see if it has installed correctly. If you are not familiar with root, use ".q" in root prompt to quit.

```
shell$ root -l

root [0] .q
```

See *http://root.cern.ch/root/Install.html* for more information on installing ROOT from source.

Appendix C

Glossary

- ***A***
  - AGKY: A home-grown neutrino-induced hadronic multiplarticle production model developed by C.Andreopoulos, H.Gallagher, P.Kehayias and T.Yang.
- ***B***
  - BY: Bodek-Yang.
- ***C***
  - COH:
  - CVS:
- ***D***
  - DarwinPorts:
  - DIS: Deep Inelastic Scattering.
- ***E***
- ***F***
  - FGM: Fermi Gas Model.
- ***G***
  - Geant4:
  - GDML:
  - GENIE: **G**enerates **E**vents for **N**eutrino **I**nteraction **E**xperiments.
  - GNuMI:
  - gevdump:
  - gevgen:

  - gmkspl:
  - gntpc:
  - gspladd
  - gspl2root:
  - gT2Kevgen:
  - gSKxstable:
  - ghAevgen:

- ***H***
  - hA:
  - hN:

- ***I***
  - IMD: Inverse Muon Decay
  - INTRANUKE: A home-grown intranuclear hadron transport MC. Intranuke was initially developed within NEUGEN for the Soudan-2 experiment by W.A.Mann, R.Merenyi, R.Edgecock, H.Gallagher, G.F.Pearce and others. Since then it was significantly improved and is now extensively used by MINOS and other experiments. Current INTRANUKE development is led by S.Dytman. INTRANUKE, in fact, contains two independent models (called 'hN' and 'hA').

- ***J***
  - JETSET:
  - JNUBEAM:
  - JPARC:

- ***K***
  - KNO: Refers to the Koba, Nielsen and Olesen scaling law.

- ***L***
  - LHAPDF: Les Houches Accord PDF Interface.
  - libxml2: The XML C parser and toolkit of Gnome (see http://xmlsoft.org).
  - log4cpp: A library of C++ classes for fleible loggingto files, syslog, IDSA and other destinations (see http://log4cpp.sourceforge.net).

- ***M***
- ***N***
  - NEUT:
  - NUANCE:

  - NuMI

- ***O***
- ***P***
  - PYTHIA:
- ***Q***
  - QEL: Quasi-Elastic.
- ***R***
  - RES: Resonance.
  - Registry:
  - ROOT:
  - RS: Rein-Sehgal
  - RSB: Rein-Sehgal-Berger
  - RSD: Remote Software Deployment Tools. A system for automated software installation developed by Nick West (Oxford).
- ***S***
  - Subversion:
  - SVN:
  - SKDETSIM:
- ***T***
- ***U***
- ***V***
- ***W***
- ***X***
  - XML: Extensible Markup Language.
- ***Y***:
  - Yum:
- ***Z***

Appendix D

The GENIE mug

GENIE development is fueled primarily by $C_8H_{10}N_4O_2$ [1] so the GENIE mug is a key component to the project success. Below, the GENIE mug in action at Rutherford Lab.

[1] Caffeine

Appendix E

Commonly used particle and status codes

Particle codes

See PDG 'Monte Carlo Particle Numbering Scheme' for a complete list.
http://pdg.lbl.gov/2008/mcdata/mc_particle_id_contents.shtml

ν_e ($\bar{\nu_e}$)	12 (-12)	p	2212	π^0	111
ν_μ ($\bar{\nu_\mu}$)	14 (-14)	n	2112	π^+ (π^-)	211 (-211)
ν_τ ($\bar{\nu_\tau}$)	16 (-16)	Λ^0	3122	ρ^0	113
e^- (e^+)	11 (-11)	Σ^+	3222	ρ^+ (ρ^-)	213 (-213)
μ^- (μ^+)	13 (-13)	Σ^0	3212	η	221
τ^- (τ^+)	15 (-15)	Σ^-	3112	η'	331
d (d)	1 (-1)	Ξ^0	3322	ω	223
u ($\bar{u}$)	2 (-2)	Ξ^-	3312	ϕ	333
s ($\bar{s}$)	3 (-3)	Ω^-	3332	η_c	441
c ($\bar{c}$)	4 (-4)	Λ_c^+	4122	J/ψ	443
b ($\bar{b}$)	5 (-5)	Σ_c^0	4112	K^0 (K^0)	311 (-311)
t ($\bar{t}$)	6 (-6)	Σ_c^+	4212	K^+ (K^-)	321 (-321)
uu ($s = 1$)	2203	Σ_c^{++}	4222	K_L^0	130
ud ($s = 0$)	2101	Ξ_c^0	4132	K_S^0	310
ud ($s = 1$)	2103	Ξ_c^+	4232	D^0 (D^0)	421 (-421)
dd ($s = 1$)	1103	Ω_c^0	4332	D^+ (D^-)	411 (-411)
su ($s = 0$)	3201			D_s^+ (D_s^-)	431 (-431)
su ($s = 1$)	3203				
sd ($s = 0$)	3101				
sd ($s = 1$)	3103				
ss ($s = 1$)	3303				
g	21				
γ	22				
Z^0	23				
W^+ (W^-)	24 (-24)				

Codes for baryon resonances

$P_{33}(1232)$; Δ^-	1114	$D_{13}(1700)$; N^0	21214
$P_{33}(1232)$; Δ^0	2114	$D_{13}(1700)$; N^+	22124
$P_{33}(1232)$; Δ^+	2214	$P_{11}(1710)$; N^0	42112
$P_{33}(1232)$; Δ^{++}	2224	$P_{11}(1710)$; N^+	42212
$P_{11}(1440)$; N^0	12112	$P_{13}(1720)$; N^0	31214
$P_{11}(1440)$; N^+	12212	$P_{13}(1720)$; N^+	32124
$D_{13}(1520)$; N^0	1214	$F_{35}(1905)$; Δ^-	1116
$D_{13}(1520)$; N^+	2124	$F_{35}(1905)$; Δ^0	1216
$S_{11}(1535)$; N^0	22112	$F_{35}(1905)$; Δ^+	2126
$S_{11}(1535)$; N^+	22212	$F_{35}(1905)$; Δ^{++}	2226
$S_{31}(1620)$; Δ^-	11112	$P_{31}(1910)$; Δ^-	21112
$S_{31}(1620)$; Δ^0	1212	$P_{31}(1910)$; Δ^0	21212
$S_{31}(1620)$; Δ^+	2122	$P_{31}(1910)$; Δ^+	22122
$S_{31}(1620)$; Δ^{++}	2222	$P_{31}(1910)$; Δ^{++}	22222
$S_{11}(1650)$; N^0	32112	$P_{33}(1920)$; Δ^-	21114
$S_{11}(1650)$; N^+	32212	$P_{33}(1920)$; Δ^0	22114
$D_{15}(1675)$; N^0	2116	$P_{33}(1920)$; Δ^+	22214
$D_{15}(1675)$; N^+	2216	$P_{33}(1920)$; Δ^{++}	22224
$F_{15}(1680)$; N^0	12116	$F_{37}(1950)$; Δ^-	1118
$F_{15}(1680)$; N^+	12216	$F_{37}(1950)$; Δ^0	2118
$D_{33}(1700)$; Δ^-	11114	$F_{37}(1950)$; Δ^+	2218
$D_{33}(1700)$; Δ^0	12114	$F_{37}(1950)$; Δ^{++}	2228
$D_{33}(1700)$; Δ^+	12214		
$D_{33}(1700)$; Δ^{++}	12224		

Codes for ions

GENIE has adopted the standard PDG (2006) particle codes. For ions it has adopted a PDG extension, using the 10-digit code 10LZZZAAAI where AAA is the total baryon number, ZZZ is the total charge, L is the number of strange quarks and I is the isomer number (I=0 corresponds to the ground state).

So, for example:

1000010010 $\rightarrow H^1$
1000060120 $\rightarrow C12$:
1000080160 $\rightarrow O^{16}$:
1000260560 $\rightarrow Fe^{56}$:

and so on.

Codes for GENIE pseudo-particles

GENIE-specific pseudo-particles have PDG codes $>=$ 2000000000.

Status codes

Description	***GHepStatus_t***	**As *int***
Undefined	*kIStUndefined*	-1
Initial state	*kIStInitialState*	0
Stable final state	*kIstStableFinalState*	1
Intermediate state	*kIStIntermediateState*	2
Decayed state	*kIStDecayedState*	3
Nucleon target	*kIStNucleonTarget*	11
DIS pre-fragm. hadronic state	*kIStDISPreFragmHadronicState*	12
Resonant pre-decayed state	*kIStPreDecayResonantState*	13
Hadron in the nucleus	*kIStHadronInTheNucleus*	14
Final state nuclear remnant	*kIStFinalStateNuclearRemnant*	15
Nucleon cluster target	*kIStNucleonClusterTarget*	16

Appendix F

Copyright Notice

(c) 2003-2009, GENIE Collaboration

For all communications:
Dr. Costas Andreopoulos
Science and Technology Facilities Council / Rutherford Appleton Laboratory
Harwell Science and Innovation Campus, Oxfordshire OX11 0QX, United Kingdom
E-mail: costas.andreopoulos@stfc.ac.uk
TEL: +44-(0)1235-445091
FAX: +44-(0)1235-446733

GNU GENERAL PUBLIC LICENSE
Version 3, 29 June 2007

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program–to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that

Appendix G

Citing GENIE

Guidelines for Fair Academic Use

The authors of GENIE endorse the MCNET guidelines for fair academic use. Users are invited to consider which GENIE components are important for a particular analysis and cire them, in addition to the main references.

Main references

All derivative works should cite:

C.Andreopoulos et al., 'The GENIE Neutrino Monte Carlo Generator', arXiv:0905.2517, submitted to NIM A.

Corresponding BibTEX entry:

```
@Article{Andreopoulos:2009rq,
   author        = "Andreopoulos, C. and others",
   title         = "{The GENIE Neutrino Monte Carlo Generator}",
   year          = "2009",
   eprint        = "0905.2517",
   archivePrefix = "arXiv",
   primaryClass  =  "hep-ph",
   SLACcitation  = "%%CITATION = 0905.2517;%%"
}
```

Appendix H

Document revision history

- Sep 07, 2009: Updated the Copyright Notice (GENIE is now distributed under GPLv3) and endorsed the MCNET guidelines for fair academic use.
- Jun 19, 2009: Document new *gT2Kevgen* option to consider only certain neutrino flux species
- Jun 10, 2009: Added diffractive event flag in gst. Updated branch descriptions. Temporarily, commented out the 'User Physics Switches' chapter and appended the list of parameters at the Physics chapter. As the scope of this manual has increased we need to rethink the chapter structure.
- Jun 07, 2009: Added section on the cross section and intranuclear rescattering reweighting (adapted from T2K internal note). Added first version of the physics section (adapted from the GENIE arXiv:0904.4043 and arXiv:0905.2517 papers)
- Jun 05, 2009: Added description of the 'numi_rootracker' format.
- May 03, 2009: Added Introduction.
- May 01, 2009: Included *BibTEX* master library and updated existing citations. Added all GENIE authors on the first page. Removed the 'GENIE collaboration' inside page. Fixed typo in the *TGraph* interpolation method (Evaluate -> Eval). Added location of example event loop skeleton file. Updated 'gst' format description (added 'ecalresp0' branch description; updated coherent event flags).
- Feb 15, 2009: Added Troubleshooting section. Modified *gT2Kevgen* to use the default nd280 units. Fix a typo in the licence.
- Feb 04, 2009: Fixed typos. Updated list of status codes. Updated example events.
- Jan 29, 2009: Added the 'Validation Tools' chapter structure. Filled-in the NuValidator part. Added the NuValidator FAQs.
- Jan 28, 2009: Added appendices on the 'genie mug' and 'commonly used particle and status codes'. Filled-in some terms in the 'glossary' appendix. Included running *gevdump* in the 'post-installation tests' section.

- Jan 17, 2009: Added sections on 'XML cross section splines file' and 'Event generation outputs'. Fixed a couple of typos.
- Jan 16, 2009: An early version of the GENIE user manual was made available online.

Bibliography

[1] C. Andreopoulos *et al.* *http://www.genie-mc.org.*

[2] S. Agostinelli *et al.*, "GEANT4: A simulation toolkit," *Nucl. Instrum. Meth.*, vol. A506, pp. 250–303, 2003.

[3] M. Bahr *et al.*, "Herwig++ Physics and Manual," *Eur. Phys. J.*, vol. C58, pp. 639–707, 2008.

[4] T. Sjostrand, S. Mrenna, and P. Skands, "A Brief Introduction to PYTHIA 8.1," *Comput. Phys. Commun.*, vol. 178, pp. 852–867, 2008.

[5] R. Brun and F. Rademakers, "ROOT: An object oriented data analysis framework," *Nucl. Instrum. Meth.*, vol. A389, pp. 81–86, 1997.

[6] H. Gallagher, "The neugen neutrino event generator," *Nucl. Phys. Proc. Suppl.*, vol. 112, pp. 188–194, 2002.

[7] D. A. Harris *et al.*, "Neutrino scattering uncertainties and their role in long baseline oscillation experiments," 2004.

[8] F. Cavanna and O. Palamara, "Geneve: A monte carlo generator for neutrino interactions in the intermediate-energy range," *Nucl. Phys. Proc. Suppl.*, vol. 112, pp. 183–187, 2002.

[9] Y. Hayato, "Neut," *Nucl. Phys. Proc. Suppl.*, vol. 112, pp. 171–176, 2002.

[10] D. Casper, "The nuance neutrino physics simulation, and the future," *Nucl. Phys. Proc. Suppl.*, vol. 112, pp. 161–170, 2002.

[11] A. Rubbia, "Nux-neutrino generator," *Talk at the 1st Workshop on Neutrino-Nucleus Interactions in the Few-GeV Region (NuINT01)*, vol. http://neutrino.kek.jp/nuint01/slide/Rubbia.1.pdf, p. 29, 2001.

[12] T. Ishida, "Charged-current inclusive distributions from k2k near detectors," 2002. Prepared for 1st Workshop on Neutrino - Nucleus Interactions in the Few GeV Region (NuInt01), Tsukuba, Japan, 13-16 Dec 2001.

[13] A. A. Aguilar-Arevalo *et al.*, "Measurement of muon neutrino quasi-elastic scattering on carbon," 2007.

[14] F. Sanchez, "Search for neutrino-induced charged current coherent pion production with carbon in a 1.3-gev wide band muon neutrino beam," *Nucl. Phys. Proc. Suppl.*, vol. 155, pp. 239–241, 2006.

[15] A. A. Aguilar-Arevalo *et al.*, “Unexplained Excess of Electron-Like Events From a 1-GeV Neutrino Beam,” 2008.

[16] E. A. Paschos, A. Kartavtsev, and G. J. Gounaris, “Coherent pion production by neutrino scattering off nuclei,” 2005.

[17] S. K. Singh, M. Sajjad Athar, and S. Ahmad, “Nuclear effects in neutrino induced coherent pion production at k2k and miniboone,” *Phys. Rev. Lett.*, vol. 96, p. 241801, 2006.

[18] L. Alvarez-Ruso, L. S. Geng, S. Hirenzaki, and M. J. Vicente Vacas, “Charged current neutrino induced coherent pion production,” *Phys. Rev.*, vol. C75, p. 055501, 2007.

[19] A. Bodek, “Muon internal bremsstrahlung: a conventional explanation for the excess electron-neutrino events in MiniBoone,” 2007.

[20] J. A. Harvey, C. T. Hill, and R. J. Hill, “Anomaly mediated neutrino-photon interactions at finite baryon density,” *Phys. Rev. Lett.*, vol. 99, p. 261601, 2007.

[21] O. Buss, T. Leitner, U. Mosel, and L. Alvarez-Ruso, “The influence of the nuclear medium on inclusive electron and neutrino scattering off nuclei,” *Phys. Rev.*, vol. C76, p. 035502, 2007.

[22] O. Benhar, N. Farina, H. Nakamura, M. Sakuda, and R. Seki, “Electron and neutrino nucleus scattering in the impulse approximation regime,” *Phys. Rev.*, vol. D72, p. 053005, 2005.

[23] J. E. Amaro, M. B. Barbaro, J. A. Caballero, T. W. Donnelly, and J. M. Udias, “Final-state interactions and superscaling in the semi- relativistic approach to quasielastic electron and neutrino scattering,” *Phys. Rev.*, vol. C75, p. 034613, 2007.

[24] C. H. Llewellyn Smith, “Neutrino reactions at accelerator energies,” *Phys. Rept.*, vol. 3, p. 261, 1972.

[25] R. G. Sachs, “High-Energy Behavior of Nucleon Electromagnetic Form Factors,” *Phys. Rev.*, vol. 126, pp. 2256–2260, 1962.

[26] H. Budd, A. Bodek, and J. Arrington, “Modeling quasi-elastic form factors for electron and neutrino scattering,” 2003.

[27] R. Bradford, A. Bodek, H. Budd, and J. Arrington, “A new parameterization of the nucleon elastic form factors,” 2006.

[28] L. A. Ahrens *et al.*, “Measurement of Neutrino - Proton and anti-neutrino - Proton Elastic Scattering,” *Phys. Rev.*, vol. D35, p. 785, 1987.

[29] D. Rein and L. M. Sehgal, “Neutrino excitation of baryon resonances and single pion production,” *Ann. Phys.*, vol. 133, p. 79, 1981.

[30] R. P. Feynman, M. Kislinger, and F. Ravndal, “Current matrix elements from a relativistic quark model,” *Phys. Rev.*, vol. D3, pp. 2706–2732, 1971.

[31] D. Rein and L. M. Sehgal, "Coherent pi0 production in neutrino reactions," *Nucl. Phys.*, vol. B223, p. 29, 1983.

[32] D. Rein and L. M. Sehgal, "PCAC and the Deficit of Forward Muons in π^+ Production by Neutrinos," *Phys. Lett.*, vol. B657, pp. 207–209, 2007.

[33] A. Bodek and U. K. Yang, "Higher twist, xi/w scaling, and effective lo pdfs for lepton scattering in the few gev region," *J. Phys.*, vol. G29, pp. 1899–1906, 2003.

[34] L. W. Whitlow, S. Rock, A. Bodek, E. M. Riordan, and S. Dasu, "A Precise extraction of R = sigma-L / sigma-T from a global analysis of the SLAC deep inelastic e p and e d scattering cross-sections," *Phys. Lett.*, vol. B250, pp. 193–198, 1990.

[35] S. G. Kovalenko, "Quasielastic neutrino production of charmed baryons from the point of view of local duality," *Sov. J. Nucl. Phys.*, vol. 52, pp. 934–936, 1990.

[36] M. Bischofberger, "Quasi elastic charm production in neutrino nucleon scattering," UMI-C821490.

[37] M. A. G. Aivazis, F. I. Olness, and W.-K. Tung, "Leptoproduction of heavy quarks. 1. General formalism and kinematics of charged current and neutral current production processes," *Phys. Rev.*, vol. D50, pp. 3085–3101, 1994.

[38] D. Y. Bardin and V. A. Dokuchaeva, "Muon energy spectrum in inverse muon decay," *Nucl. Phys.*, vol. B287, p. 839, 1987.

[39] W. J. Marciano and Z. Parsa, "Neutrino-Electron Scattering Theory," *J. Phys.*, vol. G29, pp. 2629–2645, 2003.

[40] K. S. Kuzmin, V. V. Lyubushkin, and V. A. Naumov, "How to sum contributions into the total charged-current neutrino nucleon cross section," 2005.

[41] T. Yang, C. Andreopoulos, H. Gallagher, and P. Kehayias, "A hadronization model for the MINOS experiment," *AIP Conf. Proc.*, vol. 967, pp. 269–275, 2007.

[42] Z. Koba, H. B. Nielsen, and P. Olesen, "Scaling of multiplicity distributions in high-energy hadron collisions," *Nucl. Phys.*, vol. B40, pp. 317–334, 1972.

[43] M. Derrick *et al.*, "Properties of the Hadronic System Resulting from anti-Muon-neutrino p Interactions," *Phys. Rev.*, vol. D17, p. 1, 1978.

[44] A. M. Cooper-Sarkar, "HADRON FINAL STATE RESULTS FROM BEBC," 1982. Invited talk presented at Neutrino '82, Balatonfured, Hungary, Jun 14-19, 1982.

[45] A. B. Clegg and A. Donnachie, "A DESCRIPTION OF JET STRUCTURE BY P(T) LIMITED PHASE SPACE," *Zeit. Phys.*, vol. C13, p. 71, 1982.

[46] A. Bodek and J. L. Ritchie, "Further studies of fermi motion effects in lepton scattering from nuclear targets," *Phys. Rev.*, vol. D24, p. 1400, 1981.

[47] D. S. Baranov *et al.*, "An estimate for the formation length of hadrons in neutrino interactions," PHE 84-04.

[48] R. Merenyi *et al.*, "Determination of pion intranuclear rescattering rates in muon-neutrino ne versus muon-neutrino d interactions for the atmospheric neutrino flux," *Phys. Rev.*, vol. D45, pp. 743–751, 1992.

[49] R. A. Arndt, W. J. Briscoe, I. I. Strakovsky, R. L. Workman, and M. M. Pavan, "Dispersion Relation Constrained Partial Wave Analysis of piN Elastic and piN to etaN Scattering Data: The Baryon Spectrum," *Phys. Rev.*, vol. C69, p. 035213, 2004.

[50] H. De Vries, C. W. De Jager, and C. De Vries, "Nuclear charge and magnetization density distribution parameters from elastic electron scattering," *Atom. Data Nucl. Data Tabl.*, vol. 36, pp. 495–536, 1987.

[51] S. G. Mashnik, A. J. Sierk, K. K. Gudima, and M. I. Baznat, "CEM03 and LAQGSM03: New modeling tools for nuclear applications," *J. Phys. Conf. Ser.*, vol. 41, pp. 340–351, 2006.

[52] S. G. Mashnik and A. J. Sierk, "Recent Developments of the Cascade-Exciton Model of Nuclear Reactions," *J. Nucl. Sci. Tech.*, vol. S2, pp. 720–725, 2002.

[53] A.K.Ichikawa *et al.*, "Private communication,"

[54] R.Hatcher, M.Messier, *et al.*, "Private communication,"

[55] G. D. Barr, T. K. Gaisser, P. Lipari, S. Robbins, and T. Stanev, "A three-dimensional calculation of atmospheric neutrinos," *Phys. Rev.*, vol. D70, p. 023006, 2004.

[56] G. Battistoni, A. Ferrari, T. Montaruli, and P. R. Sala, "Progresses in the validation of the FLUKA atmospheric nu flux calculations," *Nucl. Phys. Proc. Suppl.*, vol. 110, pp. 336–338, 2002.